Server Architecture Laboratory
ENTERPRISE SERVER GROUP

Next Generation I/O Link Architecture Specification:
Link Specification

Draft

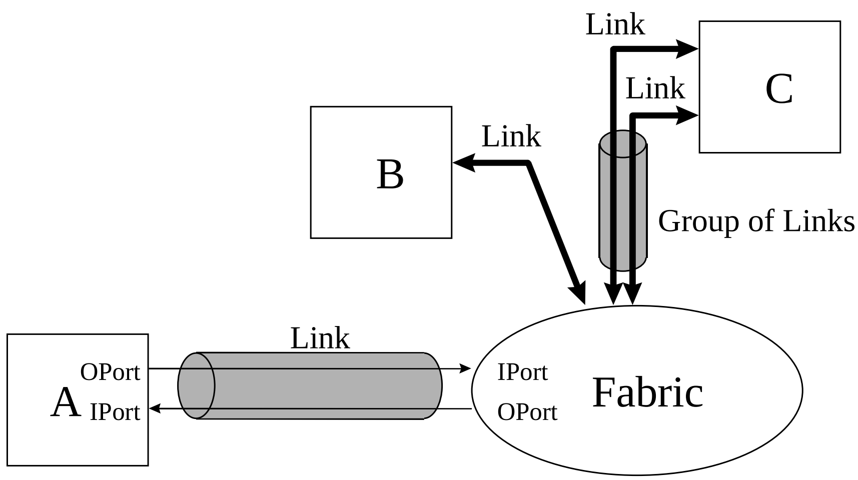
October 30, 1998

1 Introduction

A fabric is the collection of switches connecting a number of hosts and IO units. These components are connected such that two or more participants can exchange control and data. In this context, the term “link” is used to describe a bi-directional communication path between two connect points in a fabric. A fabric is then constructed out of two or more point-to-point links.

This is the Link specification. This Link specification is one of a suite of related documents which detail a new, lightweight, low latency, reliable communication protocol. Motivations and complete list of definitions and acronyms in this document are available in the Principles of Operations.

Figure 1: **Links**



The protocol detects, but does not prevent errors. To clients using the protocol, the effective behavior is that packets are not dropped, corrupted, or re-ordered due to network congestion or transient bit errors. This protocol can be used to build a reliable network.

Link protocol features:

- Large, split transaction window.
- Bandwidth scales as links are added.
- Multiple links can connect between the same two endpoints.
- An expedient, packet-based send-to-acknowledge cycle.
- Multiple checks of data correctness.
- Hardware assisted transport layer.
- Protocol is optimized for short haul links.
- Protocol can be implemented with commodity PHY serial technology.
- Protocol is optimized for correct operation, not failure cases.

1.1 Objective of this Document

This document explains the semantics and behavior for the Link architecture; a scalable, switcheable, asynchronous wire protocol.

1.2 Intended Audience

This document is written for Independent Hardware Vendors (IHV) of switch fabrics, host channel adapters, adapters for I/O such as Local or Wide Area Networks (LAN, WAN) and storage.

1.3 Scope

This document introduces an architecture for the wire interface between commodity, high performance communication hardware and transport. This document describes both areas for compliance and areas for innovation. This document uses the following industry initiatives to illustrate architectural interactions.

Gigabit Ethernet Copper Physical layer (1000BASE-CX) as example of communication hardware.

Virtual Interface Architecture (VIA) as an example of transport.

2 Transactions

The link must support three types of transactions: send, remote direct memory access (RDMA) write and RDMA read. Each transaction is either part of an acknowledged connection-oriented service, unacknowledged connection-oriented service or an unacknowledged connectionless service, each of which have different semantics described in greater detail in the Channels chapter.

In this chapter the protocol transactions are described. These transactions have a source, which initiates the transaction, and a destination, which receives the transaction. Data is transferred between source and destination through memory regions. Memory regions can be as varied as virtual memory, buffers, hardware registers, queues, or disk sectors. If the channel through which the source and destination are transacting supports an acknowledged service, the destination will respond back to the source to complete the transaction. Greater detail on attributes of channel and their service behaviors are available in the next chapter.

2.1 Send/Receive

The source transfers data to the destination. This send / receive model allows data transfer between two nodes that are managing their own buffers. At the sending end, the sending process (user or kernel) specifies the memory regions that contain the data to be sent. At the receiving end, the receiving process specifies the memory regions where the data will be placed.

When the data is correctly received at the destination, it considers the transfer complete. Across an unacknowledged service, no acknowledge (ACK) is sent back from the destination. The source considers the transfer complete when the transaction is committed to be sent.Across an acknowledged service, when the data is correctly received at the destination, an acknowledge response packet is sent back to the source. The source does not consider the transfer complete until that acknowledge (ACK) is received.

2.2 RDMA Write

An RDMA write transaction is similar in function to a send. The source transfers data to the destination. The destination does not specify where the data will be placed. An RDMA Write operation requires the source have prior knowledge of the destination memory region. The source specifies the portion of destination memory that is written to.

When writing directly into destination registered memory, the location is specified through an address and associated permissions at the destination. All memory regions must have already been registered. The destination may employ a protection mechanism, such as protection tags, to ensure the process that specified the data movement had access to that memory region.

Across a channel using an unacknowledged service, no acknowledgment response is sent back; the RDMA write descriptor is marked complete as soon as the packet is committed to be sent. Across a channel using an acknowledged service, when the data is correctly received at the destination, an acknowledge is sent back to the source. The RDMA write transfer is not marked complete until that acknowledgment is received.

2.3 RDMA Read

The source initiates the read request. The destination responds with the read response. Like the RDMA write, the RDMA read operation requires that prior to the data transfer, the source had knowledge of the location of the remote memory region, and that the destination mem-

ory region itself is enabled for RDMA read operations.

When accessing directly into registered memory, the source specifies the location of the data to be read at the destination, the length of the data being read, and a pointer to where the data is to be placed within the source memory region. The remote location of the data is specified by its virtual address and its associated memory handle. At each local node, protection mechanisms, such as tags, are provided to ensure the process that specified the data movement had access to that memory region and that the memory region has reads enabled.

RDMA read transactions only cross a channel using an acknowledged service. If servicing the read request was sucessful, no notification occurs to the destination that the request completed. A RDMA read transaction is not considered complete by the source until all the requested data is received.

2.4 Data Transfers

2.4.1 Read Response

RDMA read requests are initiated at a source. The RDMA read response packet is generated by the destination. The RDMA read response packet is returned to the source.

2.4.2 Immediate Data

In addition to data payload, RDMA write operations can convey additional information from source to destination through the immediate data field. When no immediate data is specified in an RDMA write, there is no notification at the destination that data transfer has taken place. If immediate data is specified in an RDMA write, the destination is notified on completion of data transfer.

Sends may also use the immediate data field to convey additional information. Send operations always notify the destination on data transfer. This notification behavior is not contingent on the presence of immediate data.

2.4.3 Scatter/Gather

The source may place an RDMA read response in a single, contiguous memory region or scatter it among diverse, multiple data segments. The source of an RDMA write may transfer data from one memory segment or gather it from a series of data segments. The source of a send may transfer data from one or many data segments. The destination of a send may receive data to one or many segments.

The destination of an RDMA operation (read or write) must be a single, contiguous memory region.

2.5 Ordering

Transactions are sent or received in first in, first out (FIFO) order. Transactions are never reordered. Transactions on an acknowledged service channel are acknowledged by the destination in the order they were received. These rules apply to the processing on a single channel. Ordering across multiple channels, even when sharing a common source and common destination is undefined.

2.5.1 Requests with Data

Sends and RDMA writes always complete in the order in which they are queued.

Sends do not pass Sends.

Sends do not pass RDMA writes.

RDMA writes do not pass Sends.

RDMA writes do not pass RDMA writes.

2.5.2 Requests without Data

RDMA reads do not necessarily finish moving data before Sends or RDMA writes queued after them. All requests must be submitted to memory in strict order of arrival at the destination. If completion ordering between requests without data (RDMA reads) and requests with data (RDMA writes and Sends) were strictly enforced, multiple requests with data could accumulate behind completion of a single request for a large quantity of data.

Following arrival of an RDMA read, the protocol allows for continued progress through a relaxation of the memory completion ordering rules. Data transferred through subsequent Sends or RDMA writes may be placed into destination memory before all the cells associated with an RDMA read response are pulled from destination memory and sent back to the source. This relaxation of ordering has a side-effect. When an RDMA read accesses an address which is the same as a subsequent Send or RDMA write, the RDMA read response may reflect the changes resulting from that subsequent Send or RDMA write.

RDMA reads do not pass Sends.

RDMA reads do not pass RDMA writes.

RDMA reads do not pass RDMA reads.

Sends and RDMA writes can pass RDMA reads.

2.5.3 Responses with Data

RDMA read responses always return to the source in the order in which the requests were sent.

2.5.4 Barriers

A send or RDMA write that was sent after an RDMA read request may be processed at the destination before the data associated with the read response arrives from the specified memory region. When subsequent sends or RDMA writes access the same memory region as a previous RDMA read request, the read response may reflect the changes resulting from those subsequent sends or RDMA writes.

If strict order is required, a mechanism outside the scope of the link protocol is required to guarantee the contents of memory being read have not been modified by a send or RDMA write transaction that arrived at the destination after the RDMA read request. The posting process may use a processing barrier. By setting up a barrier, the process can ensure that all work posted before the barrier was completed before processing begins on the the fenced work.

3 Channels

The protocol provides different services between channel adapters. This chapter describes channel services, their attributes, and behavior.

In this chapter the protocol transactions are described. These transactions have a source, which initiates the transaction, and a destination, which acknowledges or responds to complete the transaction. For example, during an RDMA read operation, the destination provides data to the source.

3.1 Attributes

The protocol has two general behavior attributes of channel, acknowledge and connection. Source and destination negotiate which attributes are supported on each channel at connection set up time.

3.1.1 Acknowledge

A channel which supports the acknowledge attribute is called acknowledged. This attribute is supported through hardware acknowledge of packets. A channel which does not support the acknowledge attribute is called unacknowledged. An unacknowledged channel assumes all packets are completed when sent.

3.1.2 Connection

A channel which supports the connection attribute is called connection-oriented. Destination address and destination work queue number for work on this channel are found in the source work queue context. A channel which does not support the connection attribute is called connectionless. In the case of HCA, destination address and destination work queue number are extracted from the send work descriptor.

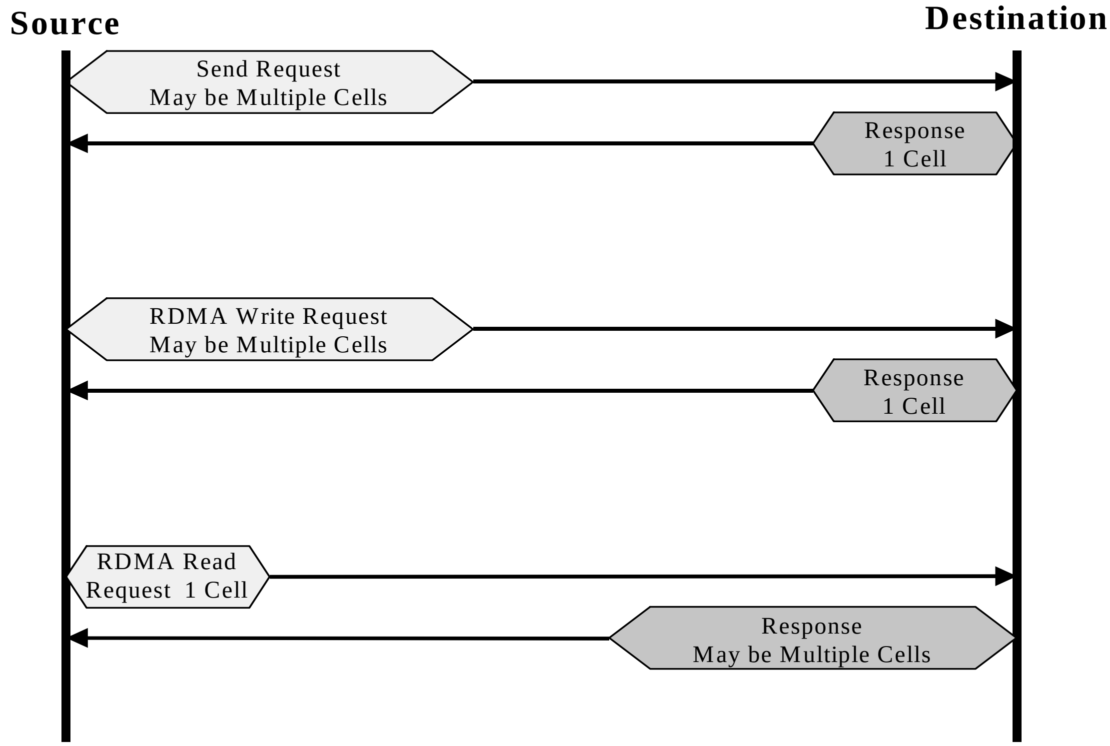
3.2 Services

Selection of channel behavior through the previously mentioned attributes affects the type of service available on that channel. Channel adaptors shall support the following three services.

3.2.1 Acknowledged Connection-oriented

Transactions initiated on an acknowledged connection-oriented service are split into a request phase and a response phase; they are not marked complete until the response is returned. The request phase of sends and RDMA writes consist of a request packet. The source may segment the packet into multiple cells. The destination must reassemble those cells into one packet. The response phase of sends and RDMA writes consists of one response packet. The request phase of RDMA reads consists of a one cell request packet. The response phase of RDMA reads consists of one response packet. The destination may segment the packet into multiple cells. The source must reassemble those cells into one packet. See Figure 2 below.

Figure 2: **Transactions on an Acknowledged Connection-oriented Service**



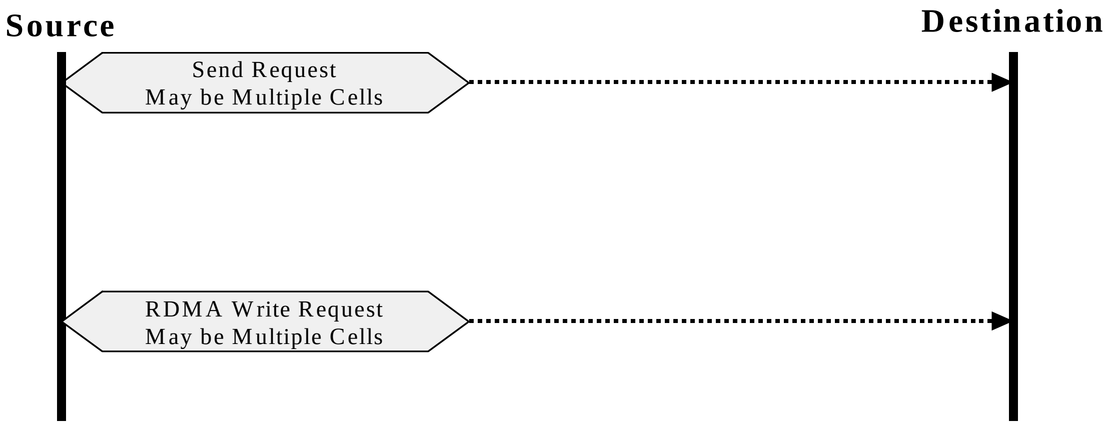
A channel which supports both the acknowledge attribute and connection attribute uses the Acknowledged Connection-oriented service. This service has the following behaviors:

- Packet sequence number (PSN) and cell sequence number (CSN) must be checked.
- Send operation must be supported. RDMA read and write operations may be supported.
- Destination must acknowledge. The source must wait for the acknowledge before completing the transaction.
- Packets may be composed of multiple cells.
- The source collects both local and remote errors.

3.2.2 Unacknowledged Connection-oriented

Send operations are supported as a request phase with no corresponding response phase. Optionally, RDMA write transactions may be supported. The request phase of sends and RDMA writes consist of a request packet. The source may segment the packet into multiple cells. The destination must reassemble those cells into one packet.

Figure 3: **Transactions on an Unacknowledged Connection-oriented Service**



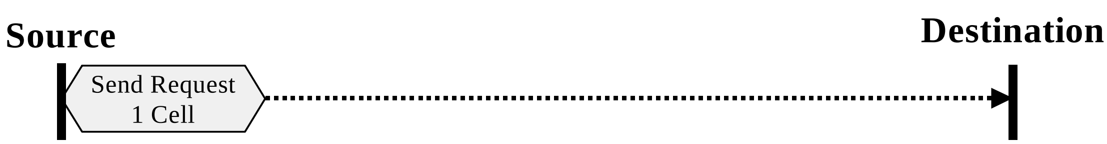
A channel which does not support the acknowledge attribute and does support the connection attribute uses the Unacknowledged Connection-oriented service.

- The packet sequence number (PSN) of a packet's first cell is not checked. If the packet is multi-cell, subsequent cells are checked for PSN consistency. Cell sequence number (CSN) is checked.
- Send operation must be supported. RDMA write operation may be supported. RDMA read operation is not supported.
- The destination does not acknowledge. The source assumes acknowledge.
- Packets may be composed of multiple cells.
- The source collects only local errors and must resolve them. The destination collects local errors and must resolve them.

3.2.3 Unacknowledged Connectionless

Send operations are supported as a request phase with no corresponding response phase. RDMA transactions are not supported. Transactions on an unacknowledged connectionless channel must always fit into one cell; they may not be segmented during transmission. Specific details and examples of how this service is used can be found in the Host Channel Adapter, Target Channel Adapter, and Fabric Services specifications.

Figure 4: **Transaction on an Unacknowledged Connectionless Service**



A channel which does not support the connection attribute and does not support the acknowledge attribute uses the Unacknowledged Connectionless service.

- Packet sequence number (PSN) is not checked. Cell sequence number (CSN) is checked and must be zero.
- Only the send operation is supported.
- The destination does not acknowledge. The source assumes immediate acknowledge.

- One cell packets, only.
- The source collects only local errors and must resolve them. The destination collects local errors and must resolve them.

3.3 Bounds

Channels have certain behaviors bounded by prior negotiation between source and destination on the conditions of connection and service.

.

Table 1: Bounding of Channel Behavior by Service

	Acknowledged Connection-oriented	**Unacknowledged Connection-oriented**	**Unacknowledged Connectionless**
Version	negotiable	negotiable	negotiable
RDMA Read Support	negotiable	N	N
RDMA Write Support	negotiable	negotiable	N
Deferred Read Amount	negotiable	N	N
Transient Error Retries	negotiable	N	N
Priority Level	negotiable	negotiable	negotiable

Version
: The source and destination agree to converse using the same version of the protocol.

RDMA Read Support
: The link must always support the RDMA read operation. A given channel using the link does not have to. Successful RDMA read operations do not notify the destination. Source and destination negotiate use of the RDMA read operation.

RDMA Write Support
: The link must always support the RDMA write operation. A given channel using the link does not have to. Successful RDMA write operations without immediate data do not notify the destination. Source and destination negotiate use of the RDMA write operation.

Deferred Read Amount
: If the channel uses RDMA reads, each end of a channel must agree on the maximum number of RDMA read requests in progress before receiving a response. The number of outstanding read requests is incremented when the request is transmitted. It is decremented when the response is completely received (or NAKed or timed out). A source may not allow this number to exceed the negotiated value. If a destination receives more requests than permitted by the negotiated value, it NAKs the errant request packet. Deferred read amount does not have to be the same on both ends of channel.

Transient Error Retries
: Protocol mechanisms attempt recovery from transient errors below the level of the transport. Source and destination agree how many retries to support before generating a transport error.

Priority Level
: The source and destination negotiate for and assign priority to the channel when the

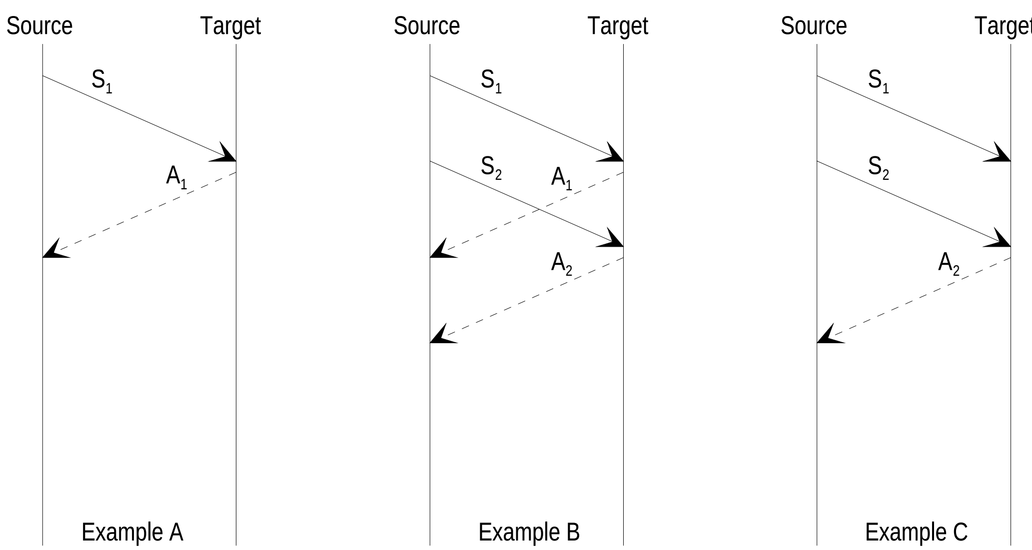
channel is created. Five levels of priority are available. Priority level must be the same on each end of channel.

4 Acknowledge

There are two types of response packets. RDMA read response packets were discussed in the previous chapter on Transactions. This chapter concerns itself with the second type of response packet called an acknowledge packet. An acknowledge packet is a special transaction with a four-byte data payload. It is only used on acknowledged connection-oriented service channels. When a request packet on a acknowledged connection-oriented channel arrives, the destination must return an acknowledge packet to the source indicating reception status. Conditions are positive acknowledge (ACK) or negative acknowledge (NAK) or no response (timeout). ACK means received without error. NAK means received with error. Timeout means the source did not receive a response within the allowed time interval.

Identification of the packet as an acknowledge is included in the opcode field of the MAC header. Reception status is included as the four byte data payload. This payload contains an error syndrome for negative acknowledges or all zeros for positive acknowledges. On a given acknowledged connection-oriented service channel, destinations must generate response packets in the same order that the request packets were sent or request packets arrived even though status may be known earlier. Unacknowledged channel services do not return response packets. The rules and guidelines discussed in this section only apply to an acknowledged connection-oriented service channel. The rules are stated in italics.

Figure 5: **Examples of Sends with Positive Acknowledgment**



4.1 Source receives an ACK

The source must wait for reception of completion status before taking any further action with a sent transaction.

When the source is an HCA, on reception of positive acknowledge completion status (ACK), the sending channel adapter writes the completion status into the appropriate send work queue descriptor. If there was a completion queue associated with the send work queue, the sending channel adapter places an entry refering to that descriptor onto its completion queue. Optionally, an event may be placed on the global event queue.

The PSN of an ACK packet identifies the original packet.

In example A, the source transmits a single Send packet and the destination responds with a

single ACK response packet. Notice that both the Send and Acknowledge packets contain the same Packet Sequence Number (PSN) which in this case is 1. Example B shows the source transmitting two Send packets. The second packet begins immediately after the first has finished and before the first was acknowledged. During the reception of the second Send, the destination returns an acknowledge for the first packet. This is possible because the links are full-duplex. After the destination receives the entire second Send packet it returns a positive acknowledge, A2.

An ACK implies a positive acknowledge of previous packets.

Example C depicts a situation where the Source operates identically as in Example B. However, the destination is unable to return Acknowledge packet A1 as in Example B. This could be due to many reasons such as the link being flow-controlled or a scheduler not choosing this work queue at this time. The destination is allowed to supersede previous positive Acknowledge packets with subsequent Acknowledge (ACK or NAK) packets. Stated differently, when the source receives an Acknowledge packet, all outstanding requests with a previous PSN receive an implied positive acknowledgment.

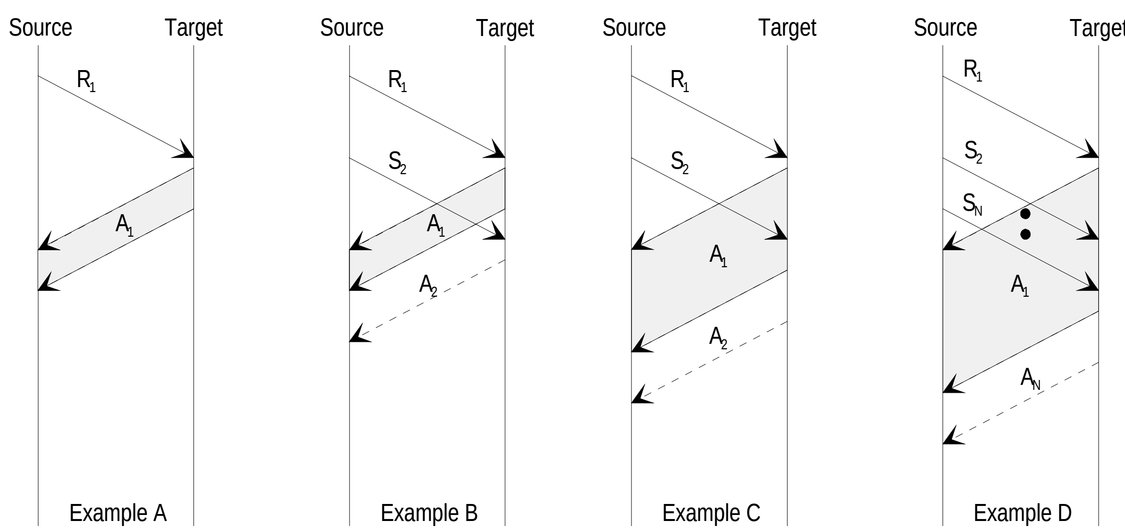
On a given work queue, the source must process outstanding descriptors in order.

For example C, the last acknowledge on this channel was A0. Packets numbered S1 and S2 are sent. The source receives an A2. The source treats this as receiving a positive acknowledge for packets S1 and S2. The source first completes S1 and then S2.

Source must ignore duplicate ACKs.

A source may retransmit a send or write due to a timeout which resulted from a delayed but not lost ACK. In this case, both ACKs may arrive back at the source. For the same rationale, a source also ignores duplicate read responses.

Figure 6: **Examples of Reads with Positive Acknowledgment.**



4.2 Destination generates an ACK

A destination must acknowledge.

When the destination is an HCA, if there was a descriptor in the receive work queue associated with the packet, when the destination generates an ACK, it also writes the completion status into the appropriate receive work queue descriptor. If there was a completion queue associated with the receive work queue, the receiving channel adapter places an entry refer-

ring to that descriptor onto its completion queue. Optionally, an event may be placed on the event queue.

In the case of a RDMA read transaction, a response with all cells and a properly formed last cell is the ACK.

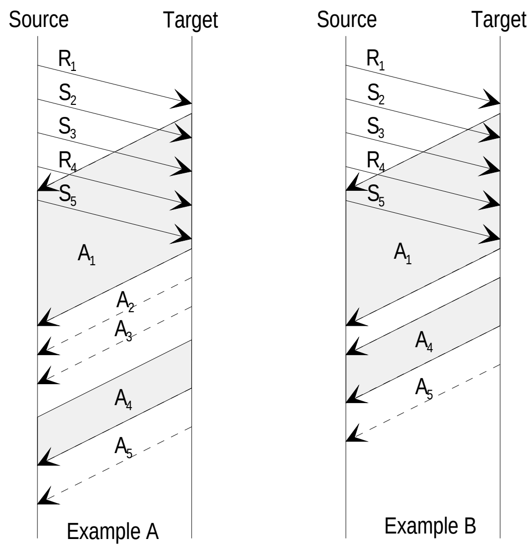
The destination must respond to a Read request by providing the requested data in a response packet. In Example A, The source issues a Read request, R1, and the destination provides the data in a response packet, A1. If the response packet is completely received without error by the source, then a positive Acknowledge is assumed.

Example B shows the source transmitting a Read request, R1, followed by a Send request, S2. In this example, the destination finishes transmitting the Read response, A1, before the Send completes. This allows the destination to explicitly acknowledge the Read request with an Acknowledge packet A1, and the Send with an Acknowledge packet, A2.

On a given work queue, destinations must acknowledge in order.

Example C is similar to Example B except that the Target takes much longer to transmit the Read response, A1. This could be because the response is very long, a scheduler is choosing between many work queues, or the link is being flow-controlled. Either way, Sends are received and processed before the original Read is completed. A destination must "remember" that S2 was received correctly and provide an Acknowledge packet, A2, after completing the Read Response, A1. Example D shows that the destination must potentially remember multiple acknowledges which accumulated during a Read response.

Figure 7: **Examples of Intermixed Sends and Reads with Positive Acknowledgement**



Special care must be given when the channel allows multiple outstanding read requests.

The Source must maintain the appropriate descriptor context information for each outstanding Read and the destination must store the address to be read and the length for each pending Read request. In Example A, while the destination is providing the Read response (and implied positive Acknowledge), A1, back to the Source, it simultaneously receives two Send requests, S2 and S3, another Read request, R4, and a third Send request, S5. The destination must wait for the first Read response, A1, to complete before it transmits A2 and A3. The destination then proceeds to respond to the second Read by transmitting A4. Only after A4 is finished, can the destination respond to the third Send request with A5. The destination was responsible for remembering that S2 and S3 occurred before R4 while S5 occurred after

R4. This allowed the destination to send back the appropriate Acknowledge packets at the correct time. In example B, the destination takes advantage of overloaded Acknowledge packets by eliminating A2 and A3. When the Source correctly receives A4, it understands that implied Acknowledges for S2 and S3 were received.

The destination always provides Acknowledge packets to Read requests, even if zero bytes of data are requested.

The destination always provides Acknowledge packets to Read requests, even if zero bytes of data are requested. It is not possible for Read requests to receive "implied" acknowledges. Only Send and Write requests can receive implied acknowledges.

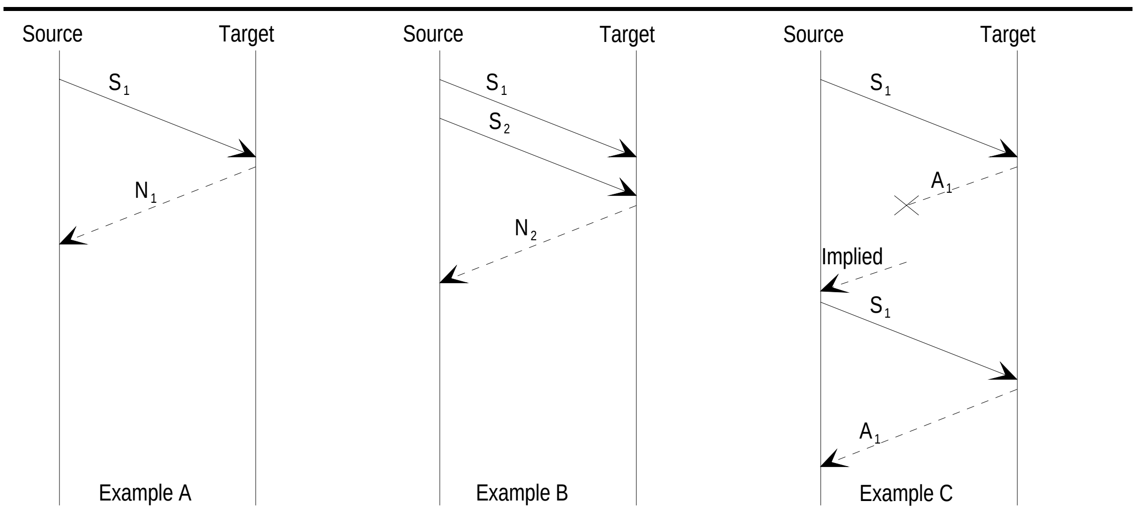
A destination is expected to respond appropriately to duplicate requests.

A destination creates a boundary at the current packet sequence number (PSN) that dictates behavior of the destination to cells or packets for this work queue sequentially previous, current, or subsequent to it. Generating an acknowledge may update the current PSN. If the PSN of the transaction to be acknowledged is an increment of one from the current PSN (wraparound is permissible) or the last of a sequential group each member of which complied with this “increment of one” rule, it qualifies as the new value of the current PSN pointer.

Any acknowledge packet is considered a positive acknowledge of all previous transactions.

Independent of acknowledge type and based on PSN, a destination ACKs sequentially previous, duplicate writes and sends, but ignores them otherwise. A destination processes sequentially previous, duplicate read requests. If the acknowledge was positive (ACK), current PSN is treated as sequentially previous. Sequentially subsequent packets for that work queue are processed, provided each is an “increment of one” from the current PSN. Sequentially subsequent packets for that work queue which do not comply with the “increment of one” rule are dropped and generate a sequence error.

Figure 8: **Examples of Sends with Negative Acknowledgment (NAK)**



4.3 Source receives a NAK

The source must pause the send work queue and generate an event when a NAK is received.

In Example A, the source issues a Send request, S1 that the destination responds to with a negative Acknowledge packet, N1. A negative Acknowledge packet can be transmitted anytime after the beginning of S1 reception. When the source receives packet N1, it resets its

Send work queue to the state that it was in before processing the Descriptor corresponding to S1. The source also pauses the channel send work queue, halting any subsequent send descriptor processing. After pausing the send work queue, a source must wait until all previous descriptors on that queue, initiated before the current transaction associated with the NAK, are completed. In this way the source guarantees the transactions complete in the order they were transmitted.

When the source detects an error locally, the behavior is the same as if it had received a NAK for that descriptor.The source does not proceed further through the error response mechanism until all previous transactions on that send work queue have been completed. The source must guarantee local events are processed in the same order as remote events would be expected to occur. If, in the process of consuming descriptors previous to the current, a descriptor with a PSN lower than the current descriptor is NAKed, the channel adapter resets current PSN pointer to the earliest NAKed descriptor.

When all outstanding transactions with PSNs lower than the current descriptor are ACKed, the source generates an event.

A NAK implies a positive acknowledge of all previous packets.

Example B shows a source transmitting two Send requests, S1 and S2. The destination chooses to respond to these requests with a single negative Acknowledge packet, N2. The source interprets this packet as a positive Acknowledge for S1 and a negative Acknowledge (NAK) for S2.

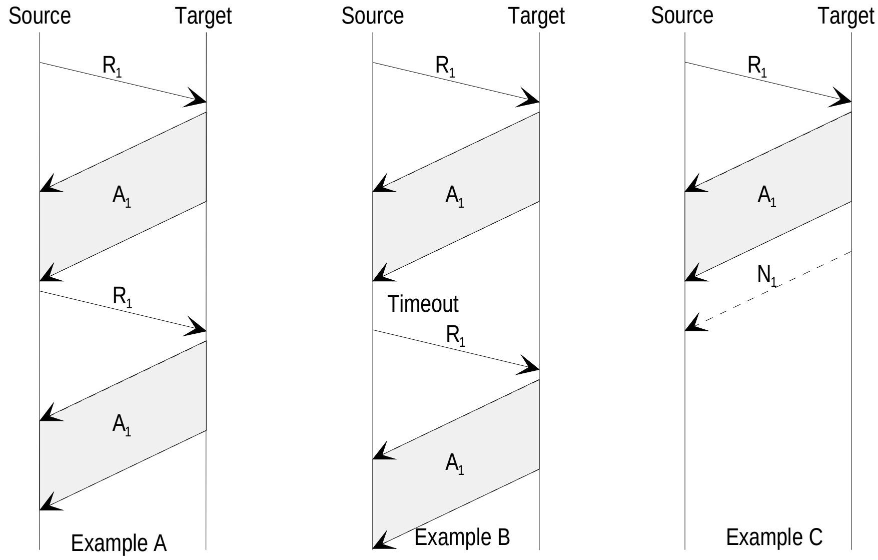
When initiating a send or RDMA write, if an error occurs, the source must transmit an improperly formed last cell.

Destinations of send or RDMA write transactions must be prepared to receive a bad packet. For errors on these transactions, the source may send a packet truncated at the point of the discovered error. This causes a mismatch in the declared size of the cell payload (length field in the MAC header, details in the Packet and Cell Format chapter) and the actual size of the cell payload (truncated by the error). In addition, the source appends an error propagation delimiter (EP character, more details in the Wire Protocol chapter) to the end of the bad cell.

Timeout is treated as a NAK by the source.

Example C shows a situation where the destination correctly responds to request S1 with a positive Acknowledge packet A1. The destination successfully completed the transaction. However, packet A1 was dropped somewhere in the fabric. Eventually, the source detects that an Acknowledge packet was not received within the timeout period. More details of timeout are to be found in the next chapter. The source must assume that the destination did not receive the packet and retransmits S1. The destination identifies the second S1 packet as having a previously acknowledged PSN and ignores all incoming cells. If the destination were to process the incoming cells then packet S1 would be received twice. The source can proceed only if the destination provides a positive Acknowledge, A1, on the last cell of the second S1.

Figure 9: **Examples of RDMA Reads with Negitive Acknowledgement (NAK)**



4.4 Destination generates a NAK

When the destination detects an error, it generates a NAK back to the source.

Duplicate requests must be acknowledged.

Previously acknowledged write and send requests must be acknowledged again. All payload data in these previously acknowledged write and send requests must be dropped. The duplicate write or send request is not processed by the destination. In contrast, the Acknowledge packet for a Read request returns all data requested. All previously acknowledged Reads must return their entire response packet. This is the only way for the Source to recognize an Acknowledge for Read requests.

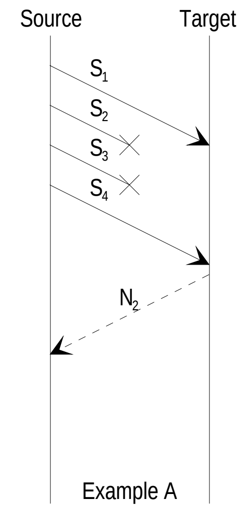
Example A above shows a destination responding to a Read request, R1. The destination responds with the entire packet, A1. One or more of the cells is dropped so the source identified the Acknowledge, A1, as negative. The source must reissue the Read request, the second R1. The destination believes that it provided a valid, positive Acknowledge with the first A1. The protocol states when a destination receives a request for a previously acknowledged packet, it retransmits the Acknowledge packet. In Example A, the entire read is performed again. Example B shows the very last cell of the Acknowledge packet A1 as dropped. The source waits for a time out period before resending its R1 request. Again, as in Example A, the destination responds a second time to what it thinks is a previously acknowledged packet.

When responding to an RDMA read, if an error occurs, the destination must respond with a NAK.

Sources of RDMA Read transactions must be prepared to receive a NAK or a data response. If the error occurs before the response packet is committed to the wire, the destination must respond with a NAK. For errors on RDMA Reads which are already committed to the wire, the destination must send back a response truncated at the point of the error. This causes a mismatch in the declared size of the cell payload (length field in the MAC header, details in the Packet and Cell Format chapter) and the actual size of the cell payload (truncated by the error). In addition, the destination must append an error propagation delimiter (EP character, more details in the Wire Protocol chapter) to the end of the bad cell. This improperly formed last cell is then followed by a NAK packet. In Example C, the destination detected a non-recoverable error such as a Translation or Protection Error. In this case, the destination immediately truncates the acknowledge packet, A1, at the fault and returns a negative Ac-

knowledge, N1, which specifically terminates the Read response.

Figure 10: **Example of Destination not Receiving all Requests**



The NAK PSN identifies the flawed, defective, or missing packet.

In Example A the source emits four Send requests, S1 through S4. The first packet, S1, is received correctly by the destination. Packets S2 and S3 are both single cell packets and neither are received by the destination. The first cell of packet S4 informs the destination that it did not receive packets S2 and S3. The destination immediately returns a negative Acknowledge packet, N2, corresponding to packet S2 because that was the first packet not received correctly.

Destinations do not timeout.

When a negative acknowledge qualifies as the current PSN, all cells or packets sequentially subsequent to the current PSN (of the NAK) are dropped until the destination gets a packet for that work queue with the “first cell” characteristic set in the MAC header and a PSN matching the current PSN or until the channel is torn down.

5 Flow Control

Several mechanisms and services are required to guarantee the protocol channel data flow behavior. This chapter describes link flow control. All services must comply with the link level protocol. Rules are stated in italics.

Five flow control characters are provided (XN0-XN4) in support of link level flow control.

Channels are established at a priority level. The priority is the same in both directions. Packets using a specific channel must all be sent at the established priority. The protocol provides five levels of priority. In ascending priority order they are 0, 1, 2, 3, and 15. The link protocol provides the different priority levels 0 through 3 to encourage and support quality of service through the fabric. The link protocol does not specify how this quality of service is to be implemented. 15 is the maximum level possible and is reserved for specific fabric management tasks. More details on these tasks is available in the Fabric Services specification.

When a source receives a XN character, it completes the transmission of any cell in process and must cease transmission of cells that are of lower priority than the XN character.

For example, when an XN3 is received, then the source is no longer permitted to transmit cells with a priority lower than 3. XN4 ceases all transmission (unless a packet is legally formed with a priority of 15, such that it is no longer subject to flow control) and XN0 enables all transmission.

A channel adapter is permitted a 100ns to respond to flow control.

Destinations transmit flow control characters to prevent loss of packets due to lack of cell buffer resources. They must be aware that activation of flow control takes time. There are five components to this time. First is the time to respond to conditions which require a flow control change. Second is the time required to complete transmission of the current cell, if any. Flow control is always emitted and responded to on cell boundaries. Third is the flight time of the XN character itself. Fourth is the recognition time by the source. Fifth is the time required to complete transmission of the current cell, if any. Destinations must ensure that they have sufficient storage for the cells received during this period. The following table shows some example calculations.

Table 2: Examples of Maximum Flow Control Interval Calculation

	30m at 1.25Gb/s	30m at 2.5Gb/s	100m at 1.25Gb/s	100m at 2.5Gb/s
potential block by cell in flight	2336[a]ns	1168ns	2336ns	1168ns
link flight time	250ns	250ns	825ns	825ns
recognize flow control (specified)	100ns	100ns	100ns	100ns
potential block by cell in flight	2336ns	1168ns	2336ns	1168ns
link flight time	250ns	250ns	825ns	825ns
total	5272ns	2936ns	6442ns	4086ns

a. This is the time on the wire for a maximum sized cell, 292 bytes.

To prevent deadlock of a network due to a lost XN character, each transmitter must emit an XN character between each cell transmitted.

This policy is enforced through the inter-cell gap mechanism which places at least one idle / flow control code-group pair between each transmitted cell. More deatils on this character control group is available in the wire protocol chapter.

6 Availability and Reliability

Several mechanisms and services are required to guarantee the protocol channel reliable behavior. This chapter describes behavior when error is detected, mechanisms of error detection, mechanisms of error notification, and types of errors.

6.1 Response to Error

There are two types of transaction errors: transmission and protocol. Protocol errors cause the correponding queue to be placed in an error state. Transactions with transmission errors, if detection is disabled, are dropped and not reported to either source or destination. The following sections discuss behavior of enabled transmission and protocol errors below API or kernel agent layers.

6.1.1 Unacknowledged Services

Protocol and enabled transmission errors cause the transaction to be marked in error where the error is detected; i.e., send side errors are reported to the source and receive side errors are reported to the destination.

6.1.2 Acknowledged Services

All protocol and enabled transmission errors are reported to the source, regardless of where they are detected. The destination never advances it's expected PSN in response to an error. The source can examine the error and determine how to proceed; this is a policy decision left to software (on an HCA source) and is not strictly part of the Link protocol.

6.1.3 Source Channel Adapter

The following are the range of source channel behaviors in response to error.. The specific response is dependent on the type of service and the type of error. More details can be found in the Host Channel Adapter and Target Channel Adapter specifications.

The source channel adapter pauses the send work queue and can perform three actions in reponse to error:

- Retry the transaction with the same PSN.
- Mark the current transaction in error and move on to the next transaction using the same PSN.
- Mark the current transaction in error and cease processing work on this queue.

Tearing down the channel may be performed in conjunction with the latter case, but this is not really part of the Link protocol. It is done via the out-of-band connection management mechanism.

6.1.4 Destination Channel Adapter

The following are the range of destination channel behaviors in response to error.. The specific response is dependent on the type of service and the type of error. More details can be found in the Host Channel Adapter and Target Channel Adapter specifications.

In the presence of error, the destination channel adapter retains the reported error and expected PSN. In response to error the destination channel adaptor will do one of the following:

- No further action required.
- Mark the current transaction in error and continue processing transactions on this queue. Increment the expected PSN.

6.1.5 Error Behavior Synchronization

The type / conditions of channel service provide the means by which consistent error behavior is established between source and destination.

6.1.6 Transient Errors, Retry, Duplicates

Source policy software may decide a reported error is transient and choose to retry the transaction. A retried transaction in error must use the same packet sequence number (PSN). If the source attempts a retry in response to a timeout, it must send the exact same ordered requests. Destinations are not required to have defined behavior if this rule is broken.

A duplicate transaction is defined as having a PSN previous to the current PSN. A destination ACKs duplicate writes and sends, but ignores them otherwise. A destination processes duplicate read requests. A source ignores duplicate ACKs. An initiator must process duplicate NAKs. A duplicate NAK is processed in the same way as the original NAK.

6.1.7 Persistent Errors

The error may be of a type which will not resolve through retry or did not resolve through several retry attempts. Through implementation of rich fabrics, fabric management services could instigate a switch to alternative routing or switchover to an alternative fabric. Polling or test of the failed route status may reveal (eventual) repairs. It is always possible to reconfigure failed routes after service.

6.2 Error Detection

Ill-formed packets must be detected and discarded. The protocol provides several methods with different error detection characteristics to reduce the opportunity for undetected error.

6.2.1 CRC-32

The Cyclic Redundancy Check (CRC) error detection technique assists the destination of a transaction to determine whether the transaction has been corrupted. The protocol uses two CRC-32 values. One is used for individual cell integrity check. A second is used for culmative packet payload integrity check. CRC encoding and decoding algorithm uses the 32-bit ethernet generating polynomial, 0x104C11DB7[1].

Cell Check

The source constructs a four-byte checksum that is a CRC function of the cell and appends it to the tail of the cell. The destination uses the same CRC function as one indication that the cell arrived correctly. The CRC covers the header, any addresses, immediate data, packet CRC, and the payload. The cell CRC does not cover cell encapsulation characters (for more details, see the wire protocol chapter).

Packet Check

Single cell packets do not have this additional packet check CRC field. In the case of multi-cell packets, the source builds a culmative checksum using all the headers, any addresses, immediate data, and the payloads of every cell of packet. The packet checksum does not include cell checksums or cell encapsulation characters. A four-byte CRC value is placed in the last cell of multi-cell packet. Like all other fields within the cell, the packet CRC value in the last cell of packet is protected by that cell’s CRC. As well as checking each cell, the destination is expected to recreate the source’s running packet checksum as an indication that the entire packet was correctly received.

1. ISO/IEC 8802-3: 1996(E) ANSI/IEEE Std 802.3, 1996 Edition, Section 3.2.8, page 15

6.2.2 Transmission Code

A binary transmission encode / decode is particularly well-suited for serial data links. The encoding scheme provides a constant DC component, regardless of data patterns. Additional redundancy also provides for clock recovery and special control characters. Bounded disparity and decode provides a form of error detection.

6.2.3 Packet and Cell Construction

Cell Payload Length

The length field in the MAC header describes the expected cell payload size. Depending on the operation, the cell payload can have up to five fields. Cell payload can include a data payload field of up to 256 bytes, 12 bytes of address / memory handle with or without a 4 byte length, 4 bytes of packet CRC, and 4 bytes of immediate data. The maximum payload size is 272 bytes. The maximum cell size is 292 bytes. The length is the sum of all fields within the cell between the header and the cell CRC.. Receiving channel adapters must compare stated cell payload length with received payload length.

Packet Sequence Number

Packets have a sequence number associated by channel. On any given channel, packets must arrive in the order transmitted. On any given channel, packets are retired in the order they were queued. There are 256 unique PSNs. Each new send work queue begins with the first packet PSN set to zero. Each subsequent packet PSN is expected to be an increment of one, modulo 256, of the previous packet's PSN.

On acknowledged channels, the number of outstanding transactions is limited to 127. Acknowledged channels partition the MAC PSN field around the current transaction PSN value. This current transaction PSN value provides a boundary which defines the 128 previous transaction PSNs and the 127 subsequent (outstanding) transaction PSNs. Any packet received which is an increment, but not an increment of one, from the current PSN is a sequence error. Transactions previous to the current PSN are not checked for sequence. On transactions of subsequent PSN, sequence is checked and reported on a packet basis, once per packet. Sequence errors result in a NAK. The NAK contains the expected PSN, not the PSN of the packet that resulted in the detection of the sequence error. If the sequence error spans multiple packets, the NAK contains the PSN of the first packet lost.

On unacknowledged channels, there is no outstanding transaction limit imposed by the protocol. The packet sequence number is a finite field and may roll over during the life of the channel. In the unlikely event that the fabric drops a group of cells whose number matches exactly the length of both packet and cell sequential counters, there is nothing which prevents an endpoint from splicing together the head of one packet with the tail of another. There is a possibility of undetected error if spliced head and tail are consistent. When the last cell of packet arrives, packet payload cyclic redundancy check is expected to detect the spliced packet error in all but a numerically insignificant number of cases.

Cell Sequence Number

Each cell within a packet also has an associated sequence number. Cell sequence numbers (CSN) provide packet integrity error coverage. Determination of error by cell sequence number on large packets can expedite delivery of a negative acknowledge to the source. This method is not perfect as the cell sequential number is a finite field and may roll over during the span of the packet or life of the channel. When the fabric drops a group of cells whose number matches exactly the length of the sequential counter, there is a possibility for undetected error. The packet payload cyclic redundancy check is also expected to detect the majority of CSN wrap errors.

Each new packet begins with the CSN set to zero. Any cell received which is an increment,

but not an increment of one, from the current CSN is a sequence error. Transactions previous to the current PSN are not checked for CSN sequence. On transactions of subsequent PSN, sequence is checked on a cell basis, but reported on a packet basis, once per packet. Sequence errors on an acknowledged channel result in a NAK. If it is only a cell sequence number error, the NAK contains the PSN of the current packet.

Authentication

Connection-oriented service channels use a specific connection for transactions. The connection is defined by four parameters, source address, source work queue pair, destination address, and destination work queue pair. Each end of a connection maintains a copy of these numbers in its local work queue pair context. Each arriving cell must contain these four parameters. The four values contained in the incoming cell's MAC header must exactly match those in the local work queue pair's stored context.

Consistency

The attributes a connection supports defines specific service available on that channel. Typically, each service will enforce a series of restrictions in terms of operations, versions, and transaction size supported. Incoming cells and packets must be both defined and consistent with the service available on that channel. This consistency of usage forms an additional redundancy check on the control portion of each received cell.

6.2.4 Watchdog Timer

The protocol transport mechanism provides a timeout mechanism for detecting the loss of packets on acknowledged class channels. The timer operates on a per send queue basis. The transport layer handles a timeout as if it had received a NAK with a PSN of one greater than the last packet successfully acknowledged.

The timer is enabled whenever there are unacknowledged packets outstanding. The timer is started when the last cell of a request packet is sent out. It is disabled (and reset to 0) when the last outstanding ACK or NAK is received. It is reset to zero whenever an ACK or NAK is received. In the case of read responses, it is reset to zero on reception of every read response cell. If the timer reaches its terminal value, then a timeout condition is indicated. The HCA specification further details an implementation of the port timer mechanism.

The fabric manager programs the timeout value for the timer on each device port. This is an eight bit unsigned value. The terminal time out is specified as the timeout value multiplied by 7.5 ms, +/- 200ppm. The actual time out must occur in no less than the terminal time and no greater than four times the terminal time. A timeout value of zero means there is no timeout in effect for this device. Guidelines for calculation of the timeout value can be found in the Fabric Services specification.

6.2.5 Probability of Undetected Error

A dangerous failure occurs when the data is corrupted in a manner that results in a flawed transmission that is internally consistent, undetected by the checksum. Unfortunately, this possibility is unavoidable. The best that can be done to minimize this possibility is to increase the amount of information and redundancy in the message and checksum.

Generally, a CRC can detect any single-burst error that does not exceed its number of check bits. Combined with the error-detection capability of encoding, CRC-32 can detect any combination of up to three additive errors, provided the beginning and end of the cell has been correctly established. A chaotic summing algorithm, where each input byte has the potential to change any number of bits in the checksum, as provided by CRC-32, makes it harder for repeating errors to be viewed as consistent.

The combination of CRC-32 and transmission encoding can be viewed as providing for the

detection of errors with a given probability. CRC-32 provides a very low a-priori probability of failure. 32 checksum bits give a 2^{-32} chance of undetected error. Application of a binary encode / decode scheme detects substantially more than half of the remaining, random error patterns. The resulting overall probability of undetected error is less than 1.2×10^{-10}. This yields an average interval between undetected error of 211,399 years when combined with the specified bit error rate, which is not to exceed 10^{-12} over serial cables at 1.25GB / sec data rates.

Cell and packet construction, packet payload CRC, higher level checks for consistency, resource allocation, and appropriate authentication also work through redundancy to reduce the undetected error rate. This contribution is not formally quantified.

6.3 Error Notification

The port status register is used to detect and report transmission errors between channel adapter media dependent and media access control hardware. The NAK packet is used to communicate protocol errors from destination to source channel adapters on acknowledged services.

6.3.1 Port Status Register

A port status register is required on all ports. Link level errors are reflected in the port status register. The link protocol does not specify the mechanisms used to control the port status register. The type of control which a channel adapter or fabric manager asserts over a port status register is defined. These errors must be enabled before they will be reported in the status register. The port status register can be reset. The port status register can be disabled from reporting an event on error. The port status bits must be configurable such that error state can be latched (persistent or “sticky”). Port status may be accumulated in counters.

A work queue using a port must check that port status register. Fabric management services are also expected to poll port status periodically. A detected port status error must generate an error notification. Transmission error event reporting must be consistent with work queue format. An example transmission error posting mechanism and event queue entry listing for each type of port status error can be found in the HCA specification availibility and reliability chapter.

The current status of the port fabric state machine is also available in the port status register. More details on the fabric state, its usage, and implications can be found in the Fabric Services specification.

6.3.2 NAK Payload

On acknowledged connection-oriented service channels, the source of a transaction is responsible for collection and resolution of all errors associated with that particular channel. These can be local errors, detected at the source, or remote errors, detected at the destination. Remote errors that occur while processing a transaction at the destination are reported back to the source through NAK packets. Later in this chapter, there is a table of protocol error definitions and error numbers. The found number in the NAK payload must be consistent with this ordinal list of errors. The cell payload of a NAK packet is four bytes long. The ordinal error report consumes bits 15-8 of that payload. The rest of the payload, bits 31-16 and 7-0 are reserved. The source must set any reserved cell field to zero. The destination is not required to check if reserved fields are set to zero.

Table 3: Nak Payload

Bits	Use	Description
31:16	Reserved	Reserved bit fields are always set to zero.
15:8	Ordinal Error Code	see Protocol Error List
7:0	Reserved	Reserved bit fields are always set to zero.

6.4 Transmission Errors

Cabled networks over distance are considered a noisy / error-introducing but not necessarily error-prone environment. Any protocol that chooses this type of media for its messages must enable the destination to determine whether the message has been corrupted.

Transmission errors are unrecoverable. The protocol does not provide forward error correction. No part of the cell can be trusted. The source can not be known. A negative acknowledge (NAK) can not be reliably returned to the source based on interpretation of any fields in a cell containing a transmission error.

6.4.1 Transmission Error List

The following is a list of errors related to hardware transmission. Disabled port status events terminate in the cell or packet being dropped with no notification to higher layers of the protocol. Enabled port status events also drop the cell or packet. Enabled port status events are reported by the first mechanism to poll the port, either a work queue intent on using the port or the fabric management service routinely polling the port.

Cells which contain the error propagation (EP) encapsulation are dropped with no error indication.

Table 4: Transmission Error List

Category	Error	Description
Port Status	Loss of sync	The port lost synchronization.
	Loss of flow control	Receive buffer overflow.
Encoding	Disparity	Running disparity on incoming characters exceeded.
	Illegal Decode	Encode does not decode to a legal character.
Cell Format	SCD alignment	SCD missing or on odd character boundary.
	EA alignment	EA missing or on even character boundary.
	ECD missing	ECD or EP character missing.
	Cell size	Cell is bigger than 292 bytes.
CRC-32	Cell Checksum	Invalid checksum.

6.5 Protocol Errors

The following is a list of errors which are acceptable at low level hardware detection level, but incorrect at a protocol level. Those errors related to cell formation are listed in the table since they may or may not be software related problems. All cells with inconsistent or undefined values in cell fields or authentication problems are dropped.

6.5.1 Protocol Error List

On an acknowledged connection-oriented service channel, all local errors at the destination are reported back to the source through NAK packets. The table lists the error number used by the NAK packet. The protocol errors are listed in a specific priority order. When a cell has multiple errors, all reportable through a NAK packet, the destination must respond with the smallest, pertinent NAK error number.

Some cause the cell or packet to be dropped, as if it had never arrived. These errors will have *dropped* in the payload column.

Table 5: Protocol Error List

Payload	Description
dropped	Destination MAC Address field does not match the incoming port address at this channel adapter.
dropped	Destination Work Queue field either does not match an active work queue or is out of the range of possible work queues at this channel adapter.
dropped	Source MAC Address field does not match context established for this channel's work queue pair.
dropped	Source Work Queue field does not match context established for this channel's work queue pair.
dropped	Priority field of the cell is inconsistent with defined channel attributes or set to a reserved value.
dropped	Version field of the cell is inconsistent defined by channel attributes or set to a reserved value.
dropped	Opcode field is inconsistent. The field is set to a reserved value, or is a change of transaction on a subsequent cell of packet, or an acknowledge on an unacknowledged channel.
dropped	Length field is out of range for opcode or does not match cell payload length.
0	**No error**. This is a positive acknowledge. There is no error in the transaction.
1	**Sequence error**. This transaction arrived out of order. PSN is greater than expected PSN (last acknowledged PSN + 1, modulo 256).CSN is other than expected CSN (last acknowledged CSN + 1, modulo 256). Packet level cyclic redundancy check has detected an error. Most likely cause is a transient transmission error.
2	**Out of Bounds error**. The transaction violates the conditions of connection. Possible causes include RDMA Read or RDMA write operation which was not permitted on this channel or the transaction has overrun the negotiated deferred read queue depth.

Table 5: Protocol Error List

Payload	Description
3	**Remote Access error**. The destination must report this specific error when an RDMA read or write transaction caused an error attempting to access the destination memory region. Possible causes include a bad address, protection violation, or insufficient rights for the requested operation.
4	**Catastrophic Resource error**. Mechanisms essential to execution of the transaction at the destination are either unavailable, missing, or unreacheable. Possible causes include a transaction which can not be marked as complete, any resource access, protection, or permission failure.
5	**Operation error**. Destination execution mechanisms are available and functioning, but the transaction could not be completed successfully. Possible causes include length, format, or memory region/buffer protection error.
6-255	**Reserved**. These values must not be used.

7 Packet and Cell Formats

Descriptors describe the data moved by transactions between two channel adapters. Transactions are implemented as single (one way) packets on unacknowledged channels or split into two packets - a request packet and a response packet on acknowledged channels. The channel protocol requires packets that contain a data payload of more than 256 bytes be segmented into two or more cells containing data payloads of 0 - 256 bytes. For connection-oriented services, segmentation may be done for the convenience of cache line alignment or expedience of a covering a memory stall. The channel protocol does not allow a connectionless service channel to segment a packet of less than 256 bytes of data payload into two or more cells.

Each cell consists of a MAC header, a cell payload and a 32 bit CRC. The payload of a cell includes data. The cell payload may contain a remote virtual address and memory handle pair. The cell payload may contain a length field specifying the number of bytes requested in a RDMA read transaction. The payload may also contain immediate data (4 bytes). The payload may be empty.

7.1 Media Access Control (MAC) Header

Each cell starts with a 16 byte MAC header. Figure 11 lists the fields left to right in order of arrival on the wire. 16-bit fields are transmitted on the wire least significant byte first. This style of data transfer is commonly called little endian.

Figure 11: **MAC Header (listed left to right in order of arrival on the wire)**

4 bits	4 bits	16 bits	16 bits	16 bits	16 bits	8 bits	8 bits	8 bits	8 bits	16 bits	8 bits
Priority	Version	Destination Address	Destination Work Queue	Source Address	Source Work Queue	Opcode	Packet Sequence Number	Reserved	Cell Sequence Number	Cell Payload Length	Reserved

The fields in the MAC header are defined as follows:

7.1.1 Priority: Bits [3:0]

Priority: The following levels of priority are defined.

Table 6: Priority Definitions

Bits [3:0]	Description
0000	Priority 0 = Lowest Priority
0001	Priority 1
0010	Priority 2
0011	Priority 3
0100-1110	Reserved
1111	Priority 15

7.1.2 Version: Bits [7:4]

Version: Specifies the version of the wire protocol. This document describes version 0001. 0000 is not a legal version.

7.1.3 Destination Address: Bits [23:8]

Destination Address: The MAC address the cell must be delivered to. This field describes a port(s) out of the switching fabric. Ports have unique MAC addresses per fabric. For systems that are constructed with two redundant sub-fabrics, two ports on a channel adapter may be assigned the same MAC address for use in a failover.

7.1.4 Dest Work Queue Number: Bits [39:24]

Destination Work Queue Number: The work queue on the receiving channel adapter that the cell is directed to. For host channel adapters, this is the same as the VI#.

7.1.5 Source Address: Bits [55:40]

Source Address: The MAC address of the port that injected the cell into the switching fabric. This field describes a port(s) into and out of the switching fabric. Ports have unique MAC addresses per fabric. For systems that are constructed with two redundant sub-fabrics, two ports on a channel adapter may be assigned the same MAC address for use in a failover.

7.1.6 Source Work Queue Number: Bits [71:56]

Source Work Queue Number: The work queue on the sending channel adapter that injected the cell in the switching fabric. For host channel adapters, this is the same as the VI#.

7.1.7 Opcode: Bits [79:72]

The OpCode field describes the operation of each cell. The encoding of the bits in this field follow:

Table 7: MAC Opcode Definitions

Bits [79:72]	Description
00000000	Send Request, First
00000001	Send Request, Middle
00000010	Send Request, Last, No Immediate Data
00000011	Send Request, Last, Immediate Data
00000100	Send Request, First and Last, No Immediate Data
00000101	Send Request, First and Last, Immediate Data
00000110	RDMA Write Request, First
00000111	RDMA Write Request, Middle
00001000	RDMA Write Request, Last, No Immediate Data
00001001	RDMA Write Request, Last, Immediate Data
00001010	RDMA Write Request, First and Last, No Immediate Data
00001011	RDMA Write Request, First and Last, Immediate Data
00001100	RDMA Read Request, First and Last
00001101	RDMA Read Response, First
00001110	RDMA Read Response, Middle
00001111	RDMA Read Response, Last
00010000	RDMA Read Response, First and Last

Table 7: MAC Opcode Definitions

Bits [79:72]	Description
00010001	Acknowledge Response (ACK or NAK)
All Others	Reserved

7.1.8 Packet Sequence Number: Bits [87:80]

Packet sequence number: The request phase of a descriptor enqueued on a VI is assigned a PSN when processing of that transaction starts. The PSN is entered into this field in the MAC header when sent out onto the wire. Request packets are assigned consecutively incrementing PSNs based on the order in which they were enqueued on the send work queue.

7.1.9 Reserved: Bits [95:88]

Reserved. Must be set to 0.

7.1.10 Cell Sequence Number: Bits [103:96]

Cell sequence number: For request packets, the sending channel adapter is required to assign consecutive CSNs for each cell of a packet. The first cell within a new packet must always be 0. If a packet contains more than 256 cells, the CSN wraps. If a transport and / or fabric error is detected by a missing CSN, that packet will be NAKed, and may be followed by retransmission of that packet. In case of error, the receive context associated with that transaction must be reset to the state prior to the start of the transaction.

7.1.11 Cell Payload Length: Bits [119:104]

The length field specifies the number of bytes contained in the cell payload. The cell payload is defined as all the bits between the MAC header and the CRC. Specifically, these bytes include a remote virtual address and memory handle pair associated with the first RDMA write cell in a packet, immediate data associated with the last cell of a send or RDMA write, a remote virtual address and memory handle pair and length associated with a RDMA read request and up to 256 bytes of data payload. Therefore, the largest valid value in the length field is 272[1]. Therefore, cells lengths may vary from 20 bytes[2] to 292 bytes[3]. Therefore the largest sized management packet is one 272 byte cell. Packets sent on a connectionless service can never be segmented into multiple cells.

7.1.12 Reserved: Bits [127:120]

Reserved. Must be set to 0.

There are two fields in the header that are reserved. All bits in these fields must be set to 0 by the source. These bits are ignored at the destination. The first reserved field (bits 95 - 88) are intended for future possible extension of the PSN field. The second reserved field (bits 127 - 120) are intended for future possible use as a next field for further encapsulation.

7.2 Cell Specification

Figure 12 below shows send cell examples. PCRC stands for “packet payload CRC”.

1. 272 Byte Cell payload (RDMA write request with immediate data): 12 bytes of address, handle pair, 4 bytes immediate data, 256 bytes of data.
2. 20 Byte Cell: 16 byte MAC header, 4 byte CRC
3. 292 Byte Cell: 16 byte MAC header, 272 bytes of cell payload, 4 bytes of CRC

Figure 12: **Send Cell Examples**

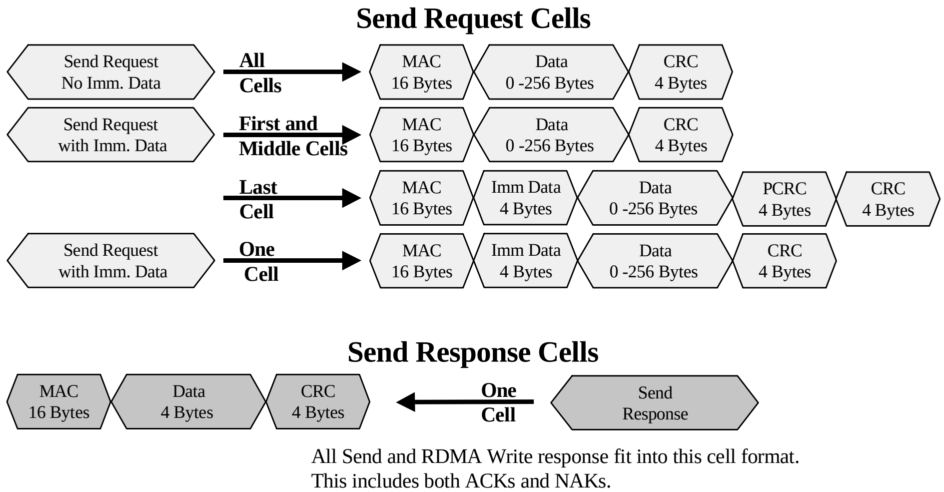
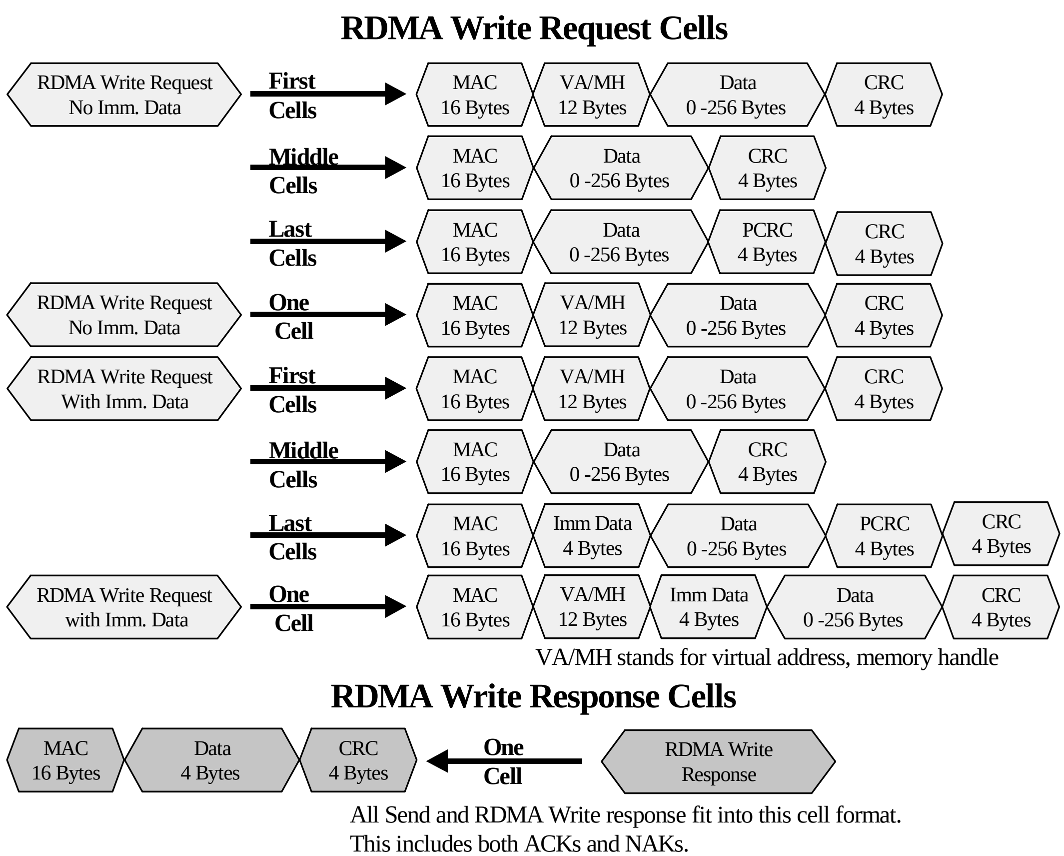


Figure 13 below shows RDMA write cell examples.

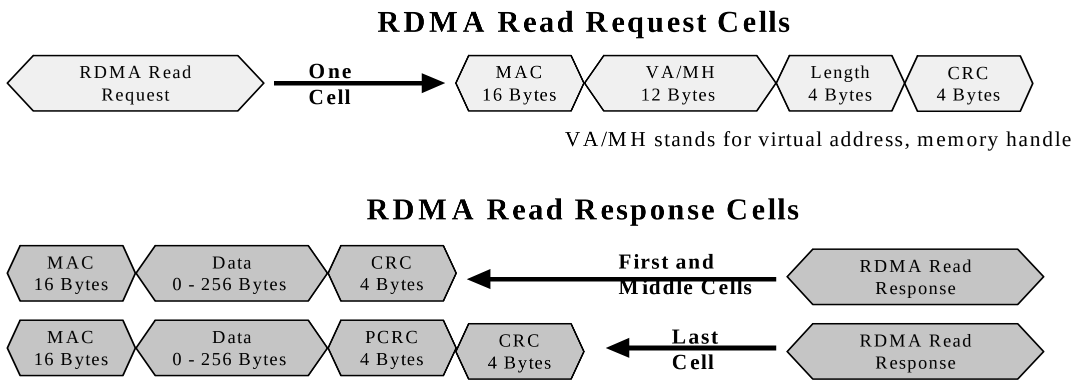
Figure 13: **RDMA Write Cell Examples**



The following diagram shows an example of an RDMA read request packet and associated response packets.

Figure 14 below shows RDMA read cell examples.

Figure 14: **RDMA Read Cell Examples**



7.3 Segmentation and Reassembly

Packets containing a data payload of more than 256 bytes are segmented into two or more cells. Packets, other than those on a unacknowledged connectionless channel, that contain a data payload of 256 bytes or less of data may be, but are not required to be segmented into two or more cells. Each cell contains a variable length data payload between 0 and 256 bytes. For packets that result in multiple cells, it is not required that any cell carry the full 256 bytes of data payload. Taking advantage of this allows data movement to be aligned on cache line boundaries.

7.4 Virtual Address/Memory Handle, Length, Packet CRC, and Immediate Data

When segmenting packets into cells, the field containing a virtual address, memory handle pair (RDMA write requests) or a virtual address, memory handle pair and read byte length (RDMA read requests) must always be placed in the first cell of a packet as one 12 byte field or 16 byte field respectively; those fields can not be split between two cells. That field must immediately follow the MAC header and precede immediate data (if present) and data (RDMA write requests) and precede the CRC (RDMA read request).

Packet payload CRC is a four byte field and must always appear in the last cell of any multi-cell packet. The PCRC field is the last field in the cell payload, immediately preceeding the cell CRC field. The PCRC field can not be split across two cells.

When segmenting packets into cells the field containing immediate data (send requests and RDMA write requests) must always be placed in the last cell as one 4 byte field; it can not be split between two cells. For cells that do not contain a virtual address, memory handle pair (sends), the immediate data field must immediately follow the MAC header and precede any data. For cells that do contain a virtual address, memory handle pair and immediate data (one cell RDMA write with immediate packet), the immediate data field must immediately follow the virtual address, memory handle pair and precede the data.

7.5 Request/Response Packets and Sequence Numbers

The split transactions only apply to reliable channels. They are initiated by the source channel adapter (initiator) with a request packet and completed by the destination channel adapter (responder) with a response packet. Send requests and RDMA write requests may be segmented into 2 or more cells by the initiating channel adapter. All of the cells within request packets have the same PSN. All of the cells within response packets have the PSN in which they are responding to. The one exception being when a response packet implicitly positively acknowledges multiple request packets; with PSNs lower than the PSN contained in the positive acknowledgment response (see below).

Each subsequent request from the same work queue contain PSNs that increment by one. All cells within a packet start with CSN = 0 and increment by one in consecutive cells.

7.5.1 Positive Acknowledgment Responses (ACK)

A positive acknowledgment (ACK) response positively acknowledges all cells with that PSN and all previous PSNs.

7.5.2 Negative Acknowledgment Responses (NAK)

If a response packet (opcode 0x11) contains any non-zero bits in the 4 byte cell payload, it is a negative acknowledgment (NAK). The (non-zero) 4 byte payload contains the error syndrome. One response packet can acknowledge more than one request packet, and NAK a request packet as well. The PSN contained in a response packet containing an error syndrome, positively acknowledges packets with PSNs less than the returned PSN (mod 256), and negatively acknowledges the packet identified by the returned PSN. Example: If the last ACK contained PSN = 223 and the next send / RDMA write response received contained a NAK of PSN = 229, that signifies that packets 228 through 224 are positively acknowledged, and PSN 229 is negatively acknowledged.

Once a packet has been NAKed by a destination, it discards all cells until it receives the first cell of a packet with a PSN <= (mod 256) to the one NAKed. Once the first cell of the packet with PSN = to the NAKed PSN is received, processing continues as normal. In all cases (whether or not a NAK has been sent), sends and writes received with a PSN <= to the greatest PSN acknowledged are acknowledged again but otherwise ignored. Read requests received with a PSN <= to the greatest PSN acknowledged are re-executed. The greatest PSN acknowledged includes PSNs implicitly acknowledged by NAKs. To NAK a RDMA read request, the responding channel adapter must stop sending back the response data (if it started) and send a response cell that contains the 4 byte error syndrome.

A NAK response can be sent at any time. Specifically, a NAK of a PSN can be sent before the last cell of the packet is received, aborting the (flawed) transfer.

7.6 Transaction, Packet, and Cell Examples

The table below is an example to help clarify the statements above.

On one send queue which is part of a reliable channel, the following transactions are enqueued:

Table 8: Request/Response Example

Type	Length (Bytes)(Decimal)	Referenced Figure
Send	640	16
RDMA write with immediate	32	17
RDMA read	90	18
RDMA read	800	19
RDMA write	399	20
Send with immediate	257	21
RDMA read	287	22

The following drawings show one compliant way to complete the above transactions. Given that each cell of a packet can be of varying length, there are many other possible ways to complete these transactions.

Figure 15: **Send Example**

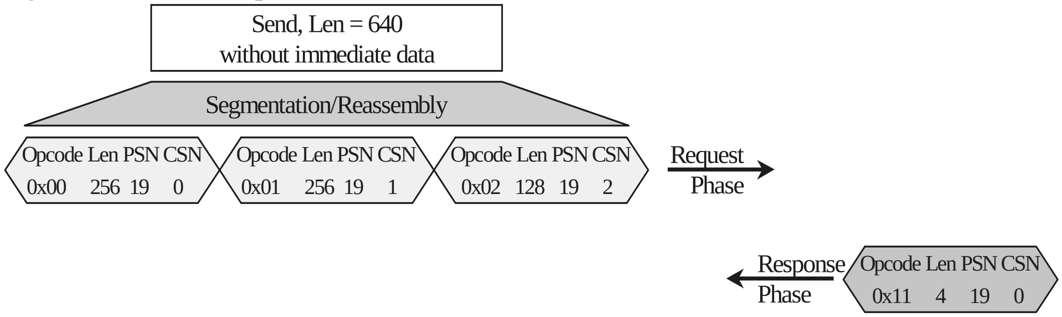


Figure 16: **RMDA Write with Immediate Data Example**

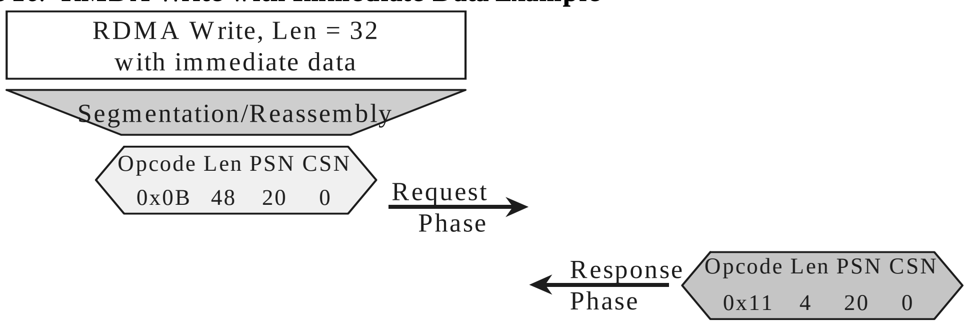


Figure 17: **RDMA Read Example**

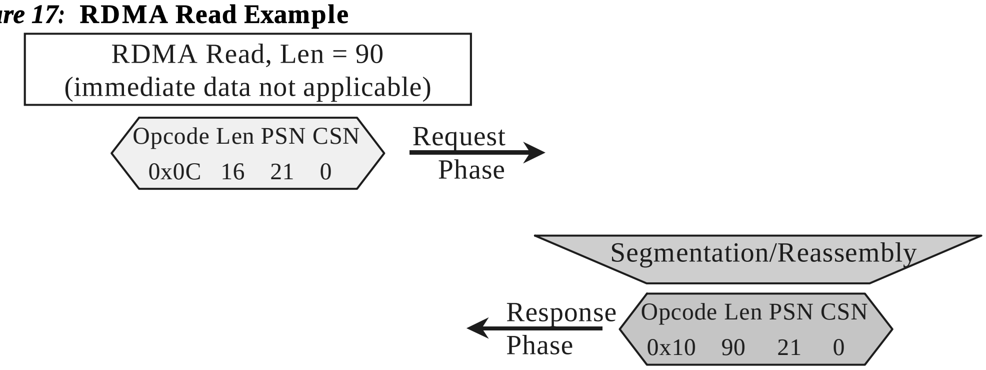


Figure 18: **RDMA Read Example**

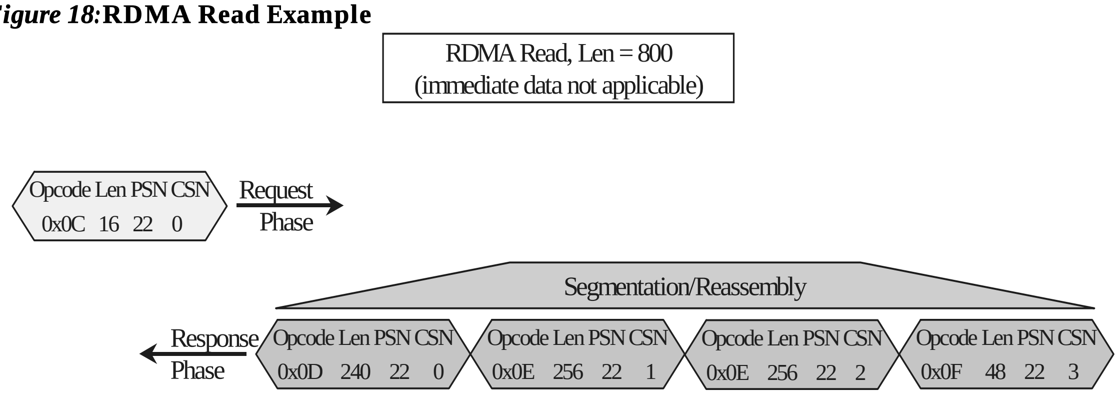


Figure 19: **RDMA Write Example**

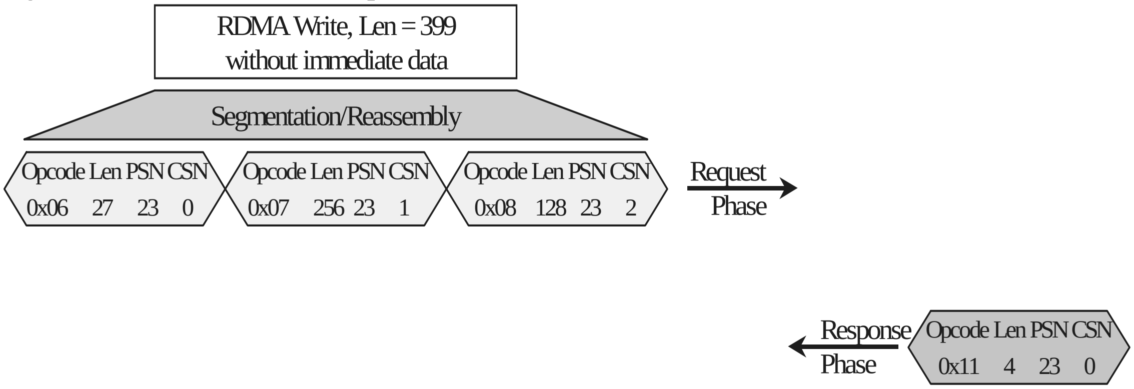


Figure 20: **Send with Immediate Date Example**

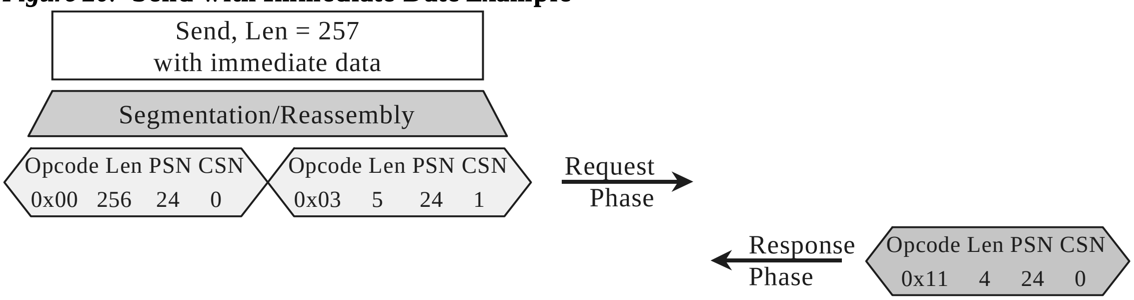
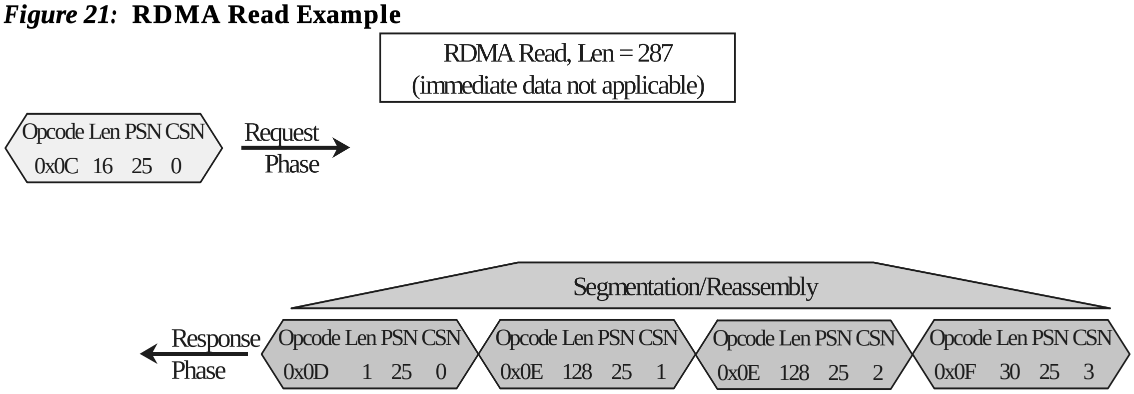
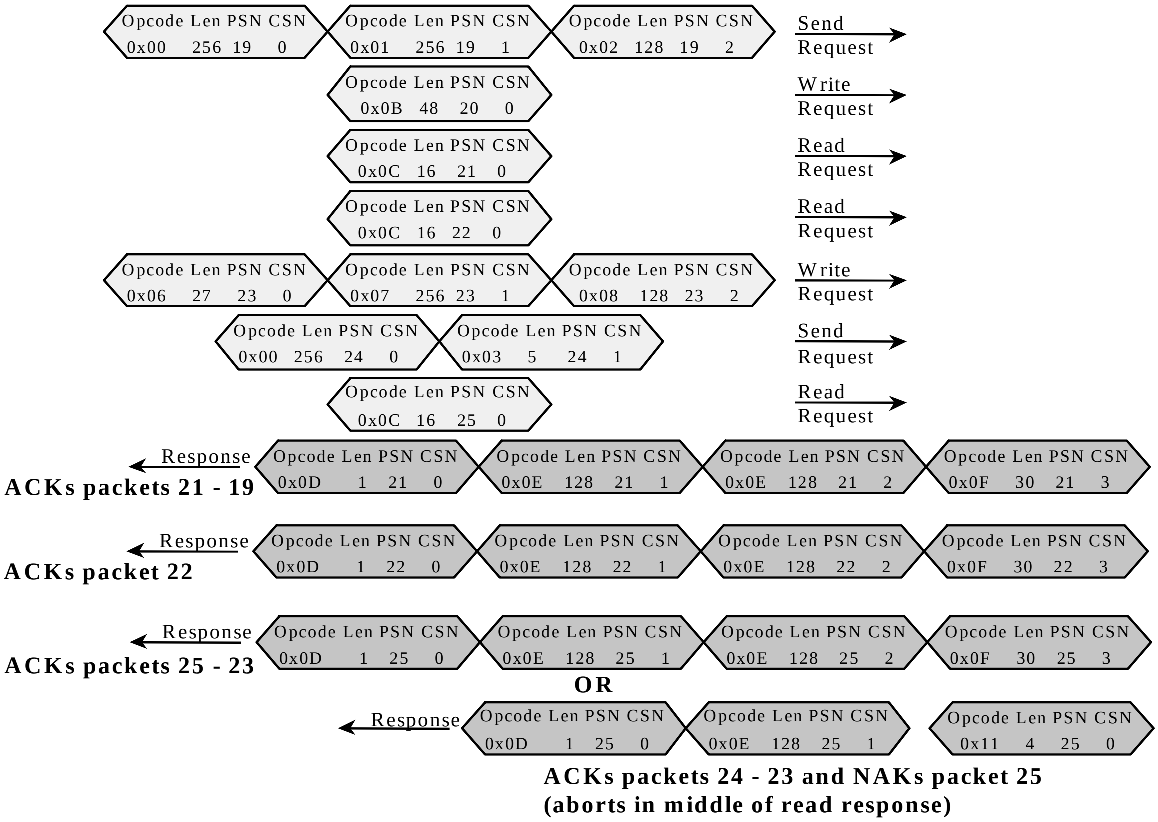


Figure 21: **RDMA Read Example**



The following drawing illustrates ACKing more that one send / RDMA write with one response packet.

Figure 22: **Multiple ACK Example**



8 Wire Protocol

The scope of this chapter is to describe in detail serial wire protocol such that a cell can be coded and properly signaled to transmit over an actual attached serial media. The wire protocol is designed to utilize standard, off-the-shelf SERDES components.

The serial wire protocol is defined to handle a full duplex Gb / s link over short haul copper or fiber optic media only. For future versions of the protocol, different standards other than serial links may be specified. This protocol may support auto-negotiation over the physical media. The physical signaling for fiber and copper media of the wire protocol is adapted from ANSI X3.230-1994(FC-PH), clauses 6 and clause 7, respectively.

The fundamentals of the serial wire protocol are based on the Gb / s Ethernet physical coding sublayer and physical medium attachment sublayer. The changes made out of Gb / s Ethernet physical layer protocol will be described in the following sections. Please refer to IEEE 802.3z specification for further information.

8.1 Physical Coding

The physical coding of the wire protocol includes the transmit, receive and synchronization processes. When communicating with the physical signaling device interface, cells are formed with a special cell delimiting code group, a coded data stream of ten bits, and proper alignment. In between the transmission and reception of cells, the special IDLE code groups or flow control code groups are formed to maintain both ends of wire alignment, synchronization and flow control.

8.1.1 Transmit Process

The transmit process keeps transmitting special code-groups in the order of the request of the synchronization process after reset or power up. When the synchronization process is at the link good state, the transmit process will start to process the incoming commands.

All code-groups generated by the transmit process will be immediately transmitted out to the physical signaling interface.

The applicable environment of this wire protocol is full duplex only, therefore there are no carrier sense and collision process required.

8.1.2 Receive Process

The receive process continuously accepts code-groups via the physical signaling interface. The receive process monitors the code-groups to detect the special code-group of start of cell-delimiter. Any receiving error detected by the receive process will cause the generation of the receive error. The cell or packet is dropped. The recieve error is recorded in the port status register, if enabled. All special flow control code-groups will be detected and decoded for the duration of the special flow control code-group.

8.1.3 Synchronization Process

The synchronization process continuously monitors the code-groups received from the physical signaling interface. The synchronization process assumes loss of synchronization state after reset or power up. Synchronization establishment is detailed later in this chapter.While it is at the link good state, the synchronization process continuously monitors the code-groups received and calculates the loss synchronization algorithm upon detecting an illegal condition against the encoding rules.

The synchronization process will perform the loss synchronization algorithm by monitoring for an illegal encoded code-group which depends on receiving on the comma code.

8.1.4 Special Characters and Code-groups

Some special characters are single characters. These are used to encapsulate a cell or for alignment purposes. Code-groups, such as idle or flow control, are made up of two character.

Table 9: Special Characters and Code-groups

Code	Definition	Number of Characters	Encoding
/I/	**IDLE**		**Correcting /I1/, Preserving /I2/**
/I1/	IDLE 1 (I1)	2	/K28.5/D22.1/
/I2/	IDLE 2 (I2)	2	/K28.5/D31.2/
	Encapsulation		
/SCD/	Start of Cell delimiter (SCD)	1	/K27.7/
/ECD/	End of Cell delimiter (ECD)	1	/K29.7/
/EA/	Even Alignment character (EA)	1	/K23.7/
/EP/	Error Propagation character (EP)	1	/K30.7/
/XN/	**Flow Control**		**Preserving /XN/**
/XN0/	Flow Control 0 (XN 0)	2	/K28.5/D2.5/
/XN1/	Flow Control 1 (XN 1)	2	/K28.5/D9.4/
/XN2/	Flow Control 2 (XN 2)	2	/K28.5/D2.6/
/XN3/	Flow Control 3 (XN 3)	2	/K28.5/D16.6/
/XN4/	Flow Control 4 (XN 4)	2	/K28.5/D2.1/

Comma Character

All special multi-code-groups defined contain a common seven bits called a comma character. This seven bit comma character is defined as 0011111 binary with positive disparity. Flow control code-groups are transmitted with the comma character to allow the receiver to do proper alignment. The comma was specifically defined for synchronization in Fibre Channel systems.

Idle Code-groups

There are two special IDLE code-groups defined. Each IDLE code-group contains a comma as its first character.

IDLE1 is defined such that the running disparity at the end of an IDLE1 transmission is opposite that of the beginning running disparity.

IDLE2 is defined such that the running disparity at the end of an IDLE2 transmission is the same as the beginning running disparity.

IDLEs are defined as a fill pattern in the inter-cell gap to establish or maintain clock synchronization. IDLEs are also used to provide the commas during inter-cell gap.

Start of Cell Delimliter Character (SCD)

A start of cell delimiter is designed to be the starting boundary of a data transmission sequence. The SCD can be transmitted after a flow control code-group.

End of Cell Delimliter Character (ECD)

An end of cell delimiter is designed to be the ending boundary of a data transmission sequence. The ECD must be transmitted after a D code-group.

Even Alignment Character (EA)

An even alignment (EA) character is designed to ensure the commas are transmitted on a code group boundary. The EA shall be transmitted only after ECD and the current position is not at the even-numbered position.

Error Propagation Character (EP)

An end of cell delimiter can be replaced by an error propagation character. It follows all the bounded behaviors of an ECD character. Its presence instead of an end of cell delimiter indicates an error was detected in the encapsulated cell previously.

Flow Control Code-groups

There are five flow control code groups defined. Each flow control code-group contains a comma as its first character. Receiving the XN0 character enables all priorities on that link, in the respective direction. The XN1 character enables priorities one and higher. It also disables priority zero. The XN2 character enables priorities two and higher. It also disables priorities one and lower. The XN3 character enables priority three. It also disables priorities two and lower. The XN4 character disables priority 3 and lower, which disables non-management traffic on the link. Priority 0-3 cells can not be dropped in the fabric except due to normal cell aging.

8.1.5 Inter-cell Gap

There is a minimum gap between transmitted cells. This is called the inter-cell gap. Besides accomodating an elastic buffering recieve port implementation, this inter-cell gap is used to provide link flow control, encapsulate a cell, and set up the comma character for character and code-group alignment.

Minimum inter-cell gap consists of six or seven characters. Either an ECD or EP character is the end of cell encapsulation. It ends the previous cell transmitted. An EA character follows the ECD character when it straddles a code-group boundary. An IDLE code-group follows next. IDLE1 is chosen if running disparity is positive. IDLE2 is chosen if running disparity is negative. More details on IDLE behavior with respect to running disparity are found later in this chapter. A flow control code-group must follow each IDLE code-group once synchronization has been established. An SCD character is the start of next cell encapsulation. It begins a new cell.

There is no maximum inter-cell gap.When there are no cells to send, transmitters are expected to transmit the IDLE / flow control code-group pair.

Setting up the Synchronizing Comma

A comma character is used by a standard “off the shelf” SERDES component for both character and character boundary alignment. These devices recognize only the positive disparity version of the comma character and expects the comma to not straddle the boundary between two transmission characters. In this protocol, the first character of the flow control code-group (K28.5) is the synchronizing comma.

If the ECD character from the end of the previous cell completed on an odd character boundary, an EA character is added to the inter-cell gap to align the flow control code-group to an even character boundary. If the ECD character from the end of the previous cell completed on an even character boundary, an EA character is omitted.

Many of the general coding and decoding rules which follow this section are derived from

the requirements of setting up this synchronizing comma.

8.1.6 Character Encoding

General Coding Rules

The transmit process starts to code groups from left to right. Bit 0 of each byte is considered the most significant bit.

Some of the special code-groups are formed by two individual code-groups, so called multi-code-group. The multi-code-group always starts with a special code group (e.g. K28.5) which mainly provides the receive synchronization and alignment function.

For each code-group (including D code-group and a special code-group) transmitted, a running disparity must be calculated. Running disparity is defined as a binary parameter with either the negative value (-) or the positive value (+). Negative means the subcode-block transmitted has more zeros than ones. Positive means the subcode-block transmitted has more ones than zeros. Each code-group is divided into two subcode-blocks.

Only the D code-groups or the special code-groups of the transmission code, selected according to the corresponding running disparity, are valid to transmit.

The special code-group Start of Cell Delimiter (SCD) and End of Cell Delimiter (ECD) are only to be transmitted legally in a pair. A cell shall be transmitted starting with SCD and ending with ECD. The Error propagation character (EP) can replace ECD on those cells for which an error has been detected.

Every SCD shall be transmitted on the even-numbered position to ensure the proper finish of multi-code-group transmission.

An Even Alignment character (EA) shall be transmitted after every ECD or EP transmission while the current position is not an even-numbered position.

One FC and one IDLE code-group must be transmitted after an ECD, EP, or EA transmission to ensure the alignment at the receiver. Some SERDES need two commas to accomplish proper alignment.

After power up and reset, the running disparity must be restored to negative by transmitting the special code-group IDLE 1 (I1).

If the running disparity is currently positive and an IDLE special code-group is required to be sent, the encoding mechanism will select I1 to restore the running disparity to the negative value.

All special code-groups for flow control symbols are multi-code-groups and are to be transmitted only in even numbered code-group position. Flow control code-groups must maintain a minimum two character interval between other flow control code-groups.

The first code-group of every multi-code-group is transmitted in an even-numbered code-group position counting from the first code-group after a reset or power-on. Subsequent code-groups continuously alternate as odd and even numbered code-groups. Note: This is designed for ease of receiving alignment.

The MAC inter cell gap (measured from de-assertion to assertion) is required to be minimum of 6 or 7 clocks (depending on final character alignment) to ensure SCD, ECD and EA.

Notes:

There are enough IDLES and EA to send in between cells to ensure the potential transmit clock variation of two end nodes.

ECD or EP character can be used interchangeably in the general coding rules.

8.1.7 Character Decoding

Receive Running Disparity

A receive code-group considered to be valid can be found in the transmission code specification, according to the current receive running disparity.

Independent of the code-group's validity, the current running disparity of the receiver is also calculated via the received code-group. The calculated disparity result of the previous received code-group will be the current running disparity for the coming code-group.

General Decoding Rules

After the synchronization process has reported the link good state, all commas shall be received on the even-numbered position.

All D code-groups must be valid from the transmission code specification according to the current running disparity.

SCD shall be received after a flow control code-group. SCD must always appear only on an even character boundary.

The special code-group ECD or EP shall be received after SCD or a D code-group.

An EA code-group shall be received after every ECD or EP transmitted while the current code-group position is not at even-numbered code-group position. EA must always appear only on an odd character boundary.

One IDLE code-groups and one flow control code-group shall be received after an ECD, EP, or an EA.

All IDLE2 code-groups shall be received at the negative running disparity position.

All comma code-group shall be received at even-numbered code-group position.

8.1.8 Synchronization and Link Detection

Synchronization happens on a full duplex link basis. Partial synchronization occurs between a transmit process and receive process on opposite sides of the link when the synchronization process receives three consecutive IDLE1 characters. During this time the transmit process is sending IDLE1 characters continuously. After the three consecutive IDLE1 characters arrive, the receiver is synchronized on a bit and character basis and can move into the Waiting_for_I2 state after sending three additional IDLE1 characters. In this state, the transmit process sends IDLE2 characters on the link and the synchronization process waits until it receives an IDLE2 character or a timeout occurs from not receiving IDLE2 characters. Timeout period id 10uS. Upon reception of an IDLE2 character, the synchronization process moves to the Link_Good state which means that the link is fully operational.

The synchronization process allows for continuous detection of the state of the link. The synchronization process tests received code-groups, and employs multiple sub-states, effecting hysteresis, to move between the link good state and Loss_Of_Sync state. The Checking I1 state allows the receiver the capability of resynchronization with the other end of the if some error condition warrants that.

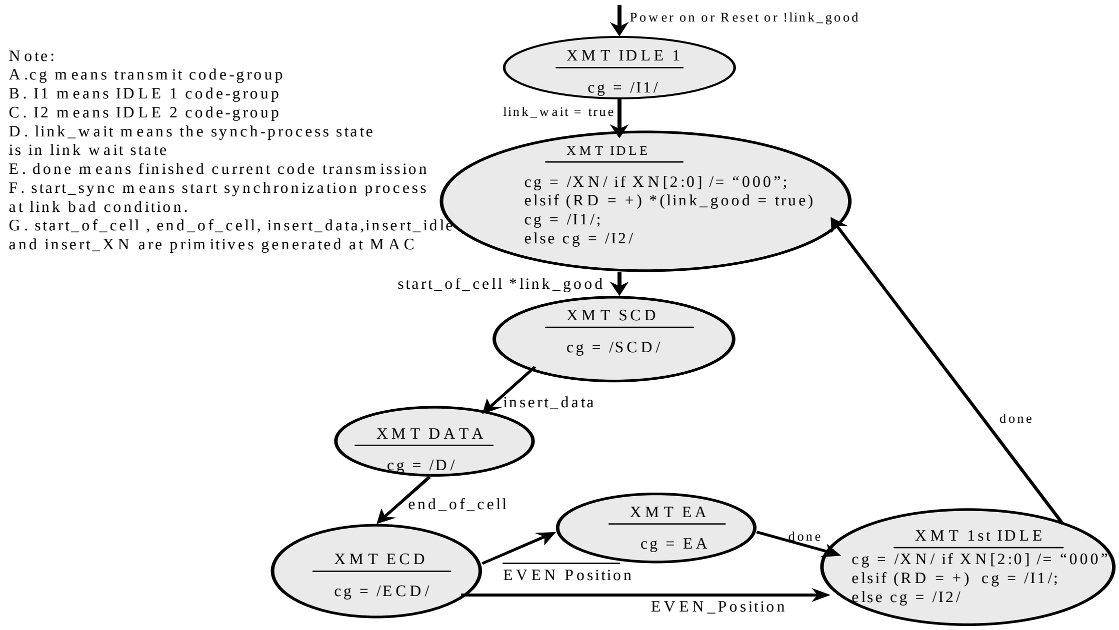
8.1.9 Transmit, Receive and Link State Machine

The following state diagrams are designed to provide a picture of the functions of the transmit and receive process. It is not intended to limit the implementations to this alone.

Transmit Function

The transmit code-group state diagram is follows. On error, the ECD character can be exchanged for the EP character.

***Figure 23:* Transmit Code-Group State Diagram**



This diagram describes the transmit cell framing sequence of the wire protocol.

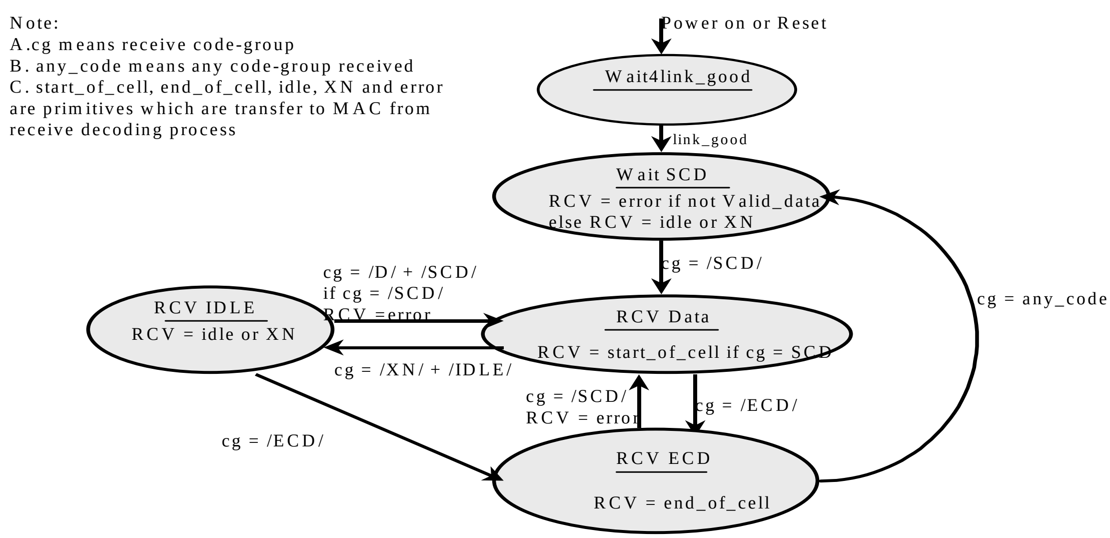
The transmit code-group state diagram also demonstrates the special flow control code group’s or IDLE code-group’s transmission.

IDLE code-groups are used to establish the link synchronization and fill in the inter cell gap.

Receive Function

On error, the ECD character can be exchanged for the EP character.

***Figure 24:* Receive Code-Group State Diagram**



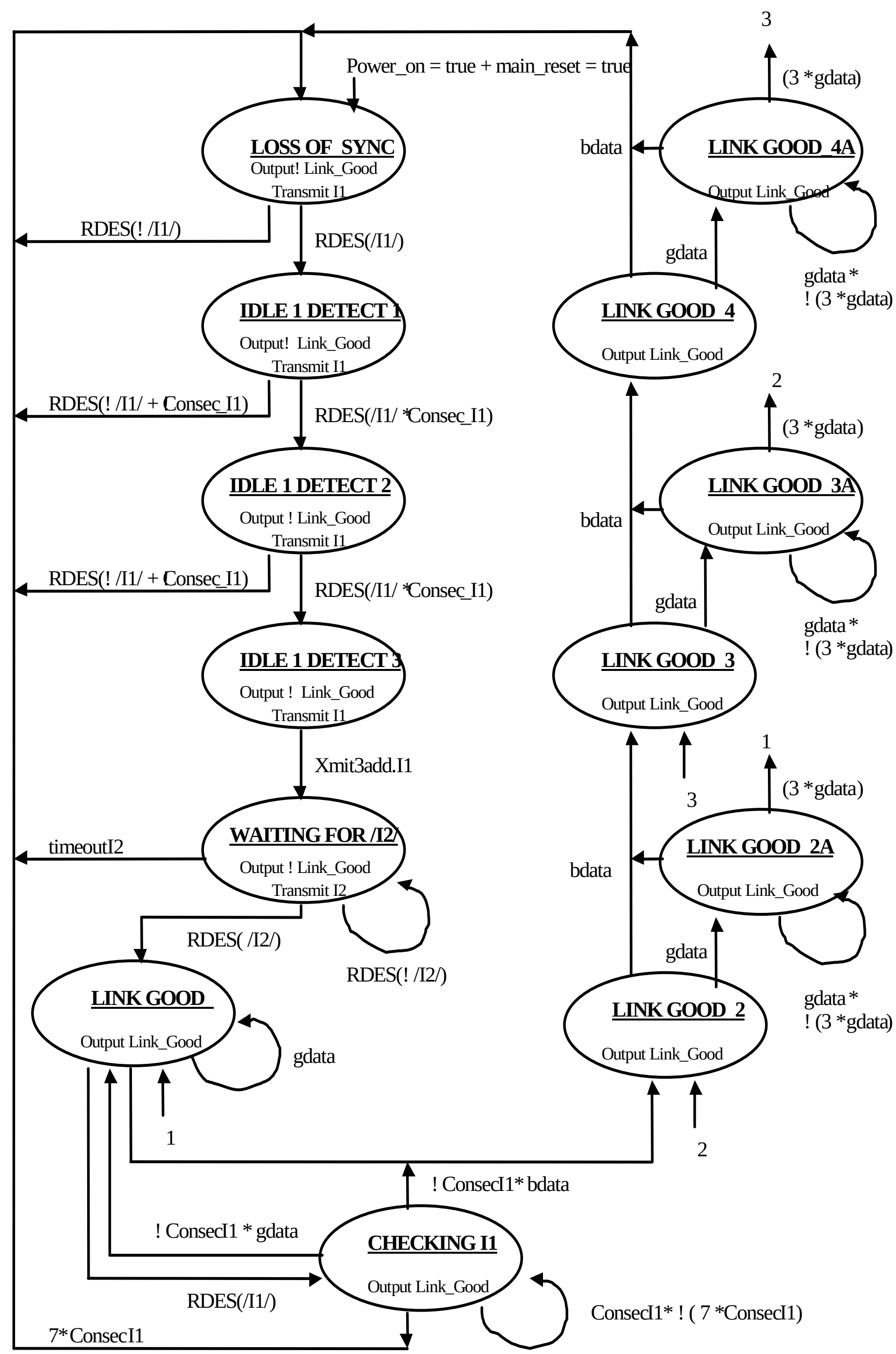
The above receive state diagram operates under the condition of proper alignment of the physical signaling block.

Each state will be evaluated when the entire 10 bit code group is received.

Link & Synchronization

The synchronization state diagram is as follows:

Figure 25: **Synchronization State Diagram**



Definition of synchronization state diagram signals

RDES

The RDES refers to a signal sent from the receive deserializer to the synchronization process. The signal is the RCV 10 bit code group.

Consec_I1

The consec_I1 signal means that a timeout has occurred on the link because the previous IDLE character received was not followed consecutively with another IDLE character as detected by the synchronization process.

Power_on

The power_on signal is true if the device that contains the PHY layer has reached its operating region. The signal is false if the device is completely powered. The signal being false is also the default.

Main_reset

The main_reset signal being true means reset the PHY layer. If the signal is false, do not reset the PHY layer.

Gdata

Gdata means that a valid 10 bit code was received.

Bdata

Bdata means that a non-valid 10 bit code was received.

TimoutI2

The timeoutI2 signal means that a timeout has occurred because an I2 character has not been received in the appropriate amount of time.

Xmit3add.I1

The Xmit3add.I1 is asserted when in the Idle_1_Detect_3 state after transmitting 3 additional /I1/ characters.

Synchronization State Machine

The synchronization state machine has two main functions. One is to monitor the receive activities. The second is to determine what to transmit in order to achieve synchronization. When synchronization is reached, we called it link good. A port can perform cell transmission and receiving at link good condition.

There are few states designed in synchronization state machines. The main states are described as following:

LOSS_OF_SYNC:

Receivers count continuous received I1 characters. If the count reaches 3, reset wait_for_I2_timer, transmit 3 more I1 characters, and go to WAIT_FOR_I2 state.

Transmitter sends continuous I1 character while in LOSS_OF_SYNC state.

WAIT_FOR_I2:

If a receiver gets a I2 Character, go to LINK_GOOD state. If the wait_for_I2_timer expired, return to the LOSS_OF_SYNC state.

Transmitter sends 3 additional I1's before transmitting continuous I2 characters.

LINK_GOOD:

The following receiver conditions cause the state change to LOSS_OF_SYNC state; reset, received continuous I1 more than 7 times, or received burst errors which caused the hysteresis study to fail.

Transmitter behaves as the normally transmit function indicated (like transmit cells, SCD, ECD, EA, I1 and I2 characters).

HYSTERESIS STUDY:

The error condition referred to is defined as any received character with illegal conditions. Illegal conditions are detected by the violation of the general coding rules designed in the earlier section.

Detecting 1 error condition will require 3 continuous good characters received to maintain LINK_GOOD state.

Detecting 2 error conditions in the interval of 4 characters received, will require 6 continuous good characters received to maintain LINK_GOOD state.

Detecting 3 error conditions in the interval of 7 characters received, will require 9 continuous good characters received to maintain LINK_GOOD state.

Otherwise:

Under detecting more errors, the hysteresis study failed, the LOSS_OF_SYNC state will be reached.

Under the condition of no error detected, the LINK_GOOD state will be maintained.

8.2 Physical Media Dependent Specification

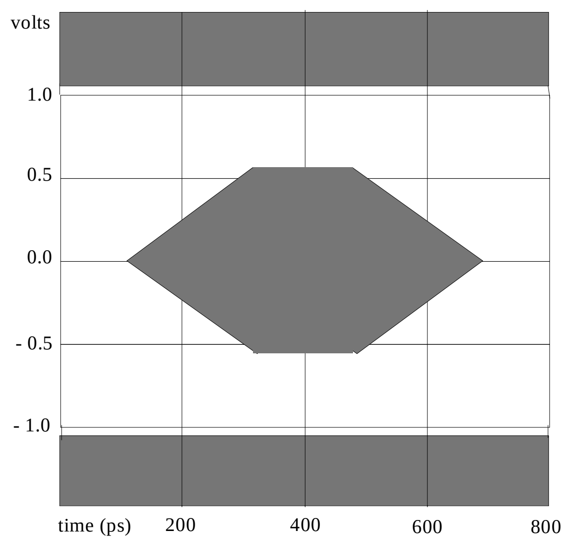
In order to form a complete specification for the wire protocol, the above physical coding and physical signaling shall be integrated with each physical media dependent separately. Since the physical coding and physical signaling are common to different physical media dependent, specification of the physical media dependent should be integrated with the above specification separately to form a complete wire protocol specification.

Appendix A —Short Haul Copper Basics

The 1000BASE_CX is designed for short-haul copper. 1000BASE_CX has a minimum operating range of 0.1 to 30 meters. Special jumper cables are required to interconnect 1000BASE_CX physical media dependent. These cables can not be concatenated to achieve longer distances. There are no changes required of the 1000BASE-CX specification for the purpose of the wire protocol specification. This appendix discusses the implications of the protocol as implemented on serial cable assemblies meeting all of the requirements for both the Fiber Channel and Gigabit Ethernet technologies. This appendix refers to and paraphrases Clause 39 of IEEE P802.3z for the convenience of the implementing engineer. This document is consistent with draft D4, dated 12/22/97, but is superseded by any further revisions of the IEEE document.

A.1 Transmitter

Figure 26: **1250 MBd Transmitter Eye Diagram**



The output driver is assumed to have PECL output levels. For all links, the output driver is AC-coupled to the media through a transmission network. Drivers have complementary outputs. Driver output amplitude must be between 1 and 2 volts. These values already include a 10% overshoot and 20% undershoot budget. An imbalance skew of 25 ps is permitted. The driver must allow less that 10% deterministic jitter at the cable connector. Transmission eye may begin transition as late as 10% of period, must open to minimum output amplitude by 30% of period, can begin closing by 70% of period, and may resolve at as early as 90% of period.

Table 10: Transmitter Characteristics

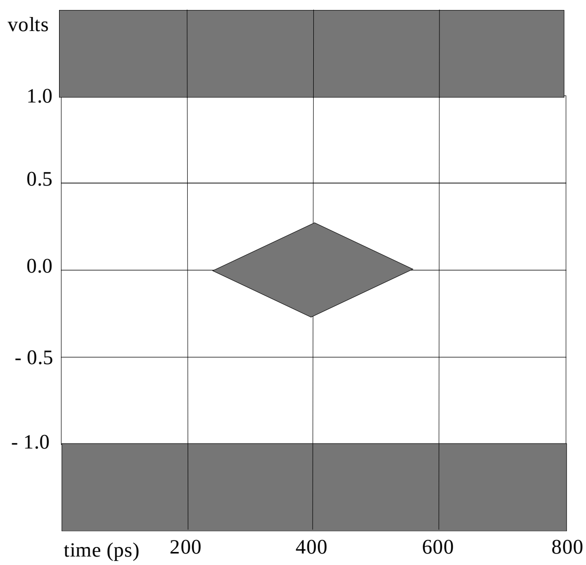
Description	Value	Unit
Type	(P)ECL	
Data Rate	1000	Mb/s
Clock Tolerance	+/-100	ppm

Table 10: Transmitter Characteristics

Description	Value	Unit
Nominal Signalling Speed	1250	MBd
Max. Differential Amplitude (p-p)	2000	mV
Min Differential Amplitude (opening)	1100	mV
Max Differential Amplitude (no power)	170	mV
Max Rise/Fall Time (20-80%)	327	ps
Min Rise/Fall Time (20-80%)	85	ps
Differential Skew	25	ps

A.2 Receiver

Figure 27: **1250 MBd Receiver Eye Diagram**



The receiver is AC-coupled to the media through the receive network. The receive network terminates the link with 150-ohm equivalent impedance +/-10 ohms. An equalizer network is optional. It can correct timing variations due to differences in propagation delay between different frequency spectrum components. It can also compensate for frequency-selective attenuation loss. Input signal amplitude must be between 0.4 and 2v. These values already include a 10% overshoot and 20% undershoot budget. Eye transition at the receiver may start as late as 30% of period, reach minimum input amplitude by 60% of period, and complete as early as 70% of period. It is recommended that the receiver budget an additional 10% for EMI-induced jitter effects.

Table 11: Receiver Characteristics

Description	Value	Units
Data Rate	1000	Mb/s

Table 11: Receiver Characteristics

Description	Value	Units
Nominal Signalling Speed	1250	MBd
Tolerance	+/-100	ppm
Min Differential Sensitivity (p-p)	400	mV
Max Differential Sensitivity (p-p)	2000	mV
Through Connection Impedance	150 +/-30	ohm
Termination Impedance	150 +/-10	ohm
Differential Skew	175	ps

A.3 Jitter

Jitter is composed of both deterministic and random components. Deterministic jitter budgetary specifications are included here to assist implementers in specifying components.

A.3.1 Reference Clock Source

One of the most important components of jitter management is the stability and accuracy of the reference source clock. Noise management is the most important attribute of a transmitter reference clock. A receiver reference clock must be frequency accurate and stable. Whether the transmitter / receiver pair is on-chip or a SERDES off-chip solution is used, it is important that the reference source clock be local, adjacent, and ideally, differential.

A.3.2 Transmitter

Transmitters introduce jitter through duty cycle distortion caused by rise and fall time driver mismatches. Use common sense guidelines of short traces treated as transmission lines in laying out gigabit signals between transmitter and cable connector. Use the PCB power and ground planes for isolation. Use controlled impedance, correctly terminated lines.

A.3.3 Receiver

Use common sense guidelines of short traces treated as transmission lines in laying out gigabit signals between cable connector and receiver. Use the PCB power and ground planes for isolation. Use controlled impedance, correctly terminated lines.

A.3.4 Interconnect Budget

This budget can be used to qualify cable and connector vendors for 1250 MBd operation. It is basically what margin remains after specification of receiver and transmitter behavior

Table 12: Interconnect Jitter Budget

Compliance Point	Total Jitter (ps)	Deterministic Jitter (ps)
Transmitter to Connector	72	16
Cable Assembly	384	208

Table 12: Interconnect Jitter Budget

Compliance Point	Total Jitter (ps)	Deterministic Jitter (ps)
Connector to Receiver	40	40

A.4 SERDES

SERDES stands for serializer / deserializer. It describes a transmitter and receiver chip deigned to perform high-speed serial data transmission. The transmitter functional partition accepts a parallel input, which it converts to bit serial, and transmits. The receiver functions partition recovers both clock and data from a serial input, converts the serial input data to parallel, detects framing, and sends the parallel data stream back to the host adaptor. The Link protocol supports off-the-shelf, standard SERDES components.

The protocol does not dictate wire speed. All SERDES vendors polled have 2500 MBd components on their near-term roadmap. The most substantial change over currently available components is in the parallel bus data rate between device and host channel adaptor. Current options include either a double width or double speed data bus.

A.4.1 General Guidelines

A SERDES adequate for the Link protocol should comply with the ANSI X3T11 Fibre Channel protocol standards and IEEE 802.3z Gigabit Ethernet applications. It should be able to operate at a minimum of 1250 MBd serial operation. It should meet or exceed setup and hold as specified by 802.3z. Link protocol requires the SERDES contain the comma character recognition and alignment circuitry as per the ANSI X3.230 FC-PH0 specification.

A.4.2 Support Component Requirements

To illustrate diverse, yet functionally adequate solutions, two different vendor's application notes have been compared. Each of the compared implementations packaged the transmitter and receiver functional units together in a 64 pin PQFP package. Maximum power dissipation ranged between 0.9 and 1.15W.

Table 13: Bill of Support Materials

	Example A	Example B
Clock	125 MHz (P)ECL	125 MHz TTL
Low Skew Buffer	clock distribution between NIC and SERDES	clock distribution between NIC and SERDES
Inductors	2x Murata BLM32A06	N/A
Loop Filter(s)	internal	2x 0.1 uF X7R ceramic
Decoupling (large)	8x 0.1uF X7R ceramic	6x 0.1 uF X7R ceramic
Decoupling (small)	8x 100 pF X7R ceramic	N/A

A.5 Serial Cable Assemblies

A.5.1 Cable

Maximum bandwidth is 1.25 Gbits/s for longer distances above 10 meters at this time without equalization. Maximum bandwidth without equalization of 10 meters is 2.5 Gbits/s. Maximum length without equalization at 1.25 Gb/s is about 25 meters. There is a standard 8 position (pins) in connector. There are 4 data signals in the cable. There are two transmit and receive differential signals in the quad-ax construction which is most popular. The outer-shielding on the cable is usually overlapped foil facing braid with 85 or 95% coverage. The other 4 signals are optional. They are usually used for power, ground, module fault detect, and optical power loss detect which are used for opto-electronic cable/module converter applications.

The Link protocol specifies copper cable capability to a maximum of 30 meters in length. Clause 39 of IEEE 802.3z/draft 4 specifies that cable shields are grounded through the bulkhead connector shell on both the transmitter and receiver ends. The protocol would prefer to chassis ground on one side only, but defers to the specification. The cable connector is the plug or male. The bulkhead connector is the receptacle or female connector. Maximum attenuation can not exceed 0.25dB/meter. The cable is allowed an impedance tolerance of 10% and may exhibit as much as 175 ps of differential skew.

Equalized vs. Unequalized cable assemblies

Unequalized HSSDC cable assemblies at the 1.25 Gb/s range appear from preliminary tests to easily go to 20 or 25 meter lengths without needing equalization circuitry. Equalized cable assemblies can come in two forms. First, the circuitry could be an active equalizer chip located in between the Transceiver chips and the cable assemblies. This is the more costly approach, however, this equalizer extends the cable to the 60 meter length. Another problem associated with this is someone could not plug a 1 meter or 5 meter length cable into the equalizer circuit and get it to work. The equalization circuitry would need to know the length of the cable. The other way is to place a simple passive equalizer (RC type circuit) in the male connector on the receive strands. This method is more widely used and simpler. However, longer distances than 35-40 meters become a problem. Peak to peak voltage is reduced with an RC circuit equalizers. Other methods of putting equalization techniques right in the transceivers are being explored by companies to extend the cable maximum length into the 100 meter category.

Table 14: Cable Strand Gauge Length

Cable AWG	Length (m) at 1250 MBd
30	14
28	17
26	20
24	26
22	30

A.5.2 Connector

Connection is made with Style-2 Fibre Channel balanced cable connectors. This connector has also been ratified for use with Gigabit Ethernet and is under strong consideration by the

1394b committee. It is an 8-pin plug and receptacle set. Only pins 1,3,6, and 8 are populated. These pins are connected to transmit +, transmit -, receive -, and receive + respectively. The plug shell is connected to cable shield. The receptacle shell is connected to bulkhead ground. Loss due to each connector should be less than 0.25dB.

Table 15: Style-2 Connector Contact Assignment

Contact	Signal
1	Transmit +
2	Reserved
3	Transmit -
4	Reserved
5	Reserved
6	Receive -
7	Reserved
8	Receive +

A.5.3 Assembly Electrical Characteristics

A cable assembly includes a continuous shielded, balanced cable terminated at each end with a polarized, shielded plug. A cable assembly may include an equalizer network in order to meet specification and signal quality requirements. The equalizer shall need no adjustment. All cable assemblies containing such circuits shall be marked with information identifying the specific designed operational characteristics.

Table 16: Cable Assembly Characteristics

Description	Value	Unit
Max Differential Skew	150	ps
Connection Impedance	150 +/-30	ohm
Cable Impedance	150 +/-10	ohm
Max Round Trip Delay	253	ns
Max Attenuation	8.8	dB

Appendix B — Bandwidth Aggregation

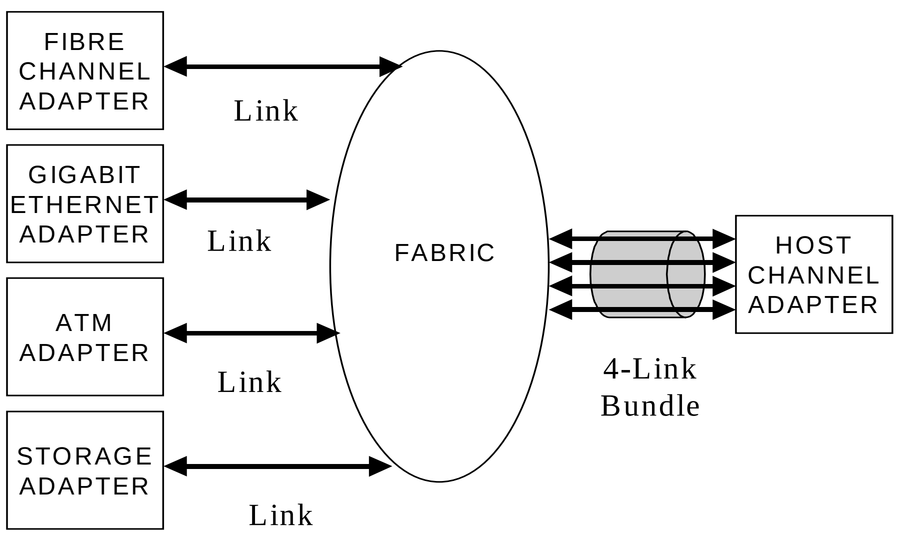
In some fabric configurations, bandwidth demands may exceed the capabilities of a single link. Although it is preferred to overcome bandwidth constraints though topology, symmetric workload distribution, and / or higher link speed; Link protocol provides method called Multiple Link Expansion (MLX) to remove the single-link bandwidth bottleneck and also reduce serial latency. Multiple links can connect between two devices in the fabric. Implementations which support bundling must be fully compatible to non-bundled implementations. MLX is a work in progress vision for next generation implementations of the Link protocol.

B.1 MLX

The ability of a single host to service mutliple targets should be limited by the server's bandwidth not bottlenecked by the link's bandwidth. The Link protocol proposes a solution to this problem through a feature called MLX. Stated simply, adding bandwidth is as easy as adding links. This improves speed matching and reduces serial latency on large packets between these high-performance channel adapters.

Links are restricted to power of two bundling. This restriction simplifies the control logic required to implement bundling. In addition to implementation simplification, this restriction is functionally consistent with the two most common anticipated uses for bundling, link speed matching and tree topology branching.

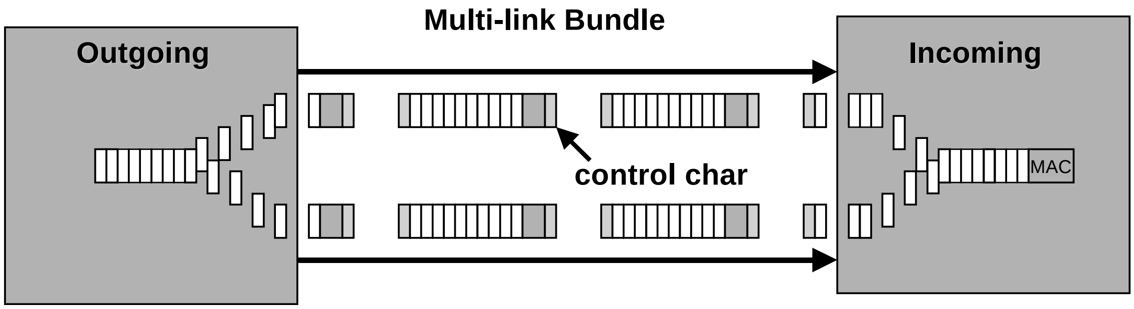
Figure 28: **Bandwidth Aggregation at the Point of Greatest Bandwidth Demand**



The other side of bandwidth improvement is the reduction in latency of large packets over bundled links. The following figure demonstrates the character-wise striping of cell data payload across multiple links used by MLX. Control code-groups are repeated on each link.

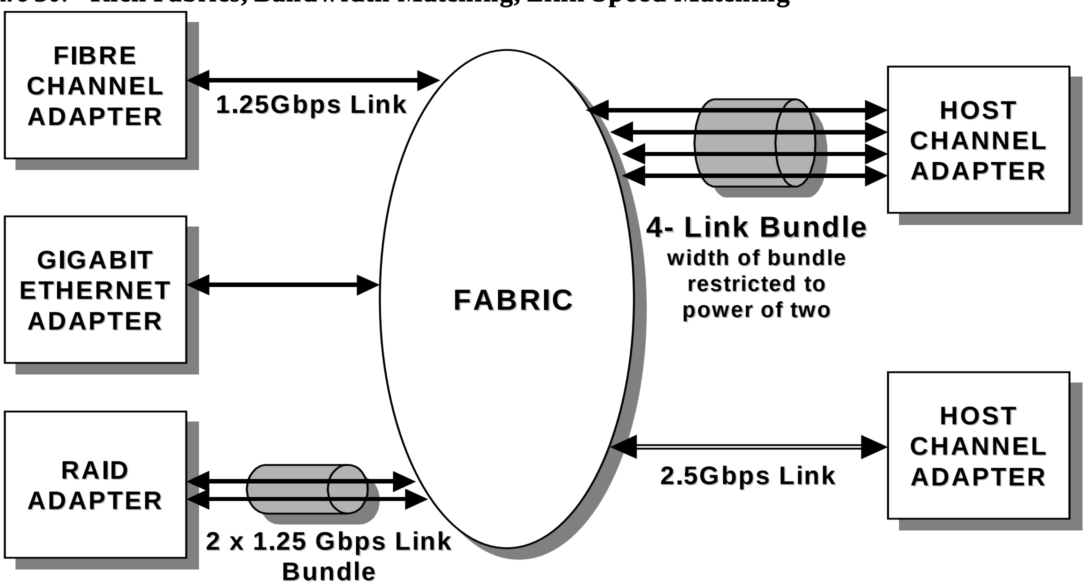
Cells are character striped across bundled links. The order in which the characters are striped across bundled links maintains a configured ordering explained in more detail in the "how its done" portion of this chapter.

Figure 29: **A Packet Striped Across Two Links**



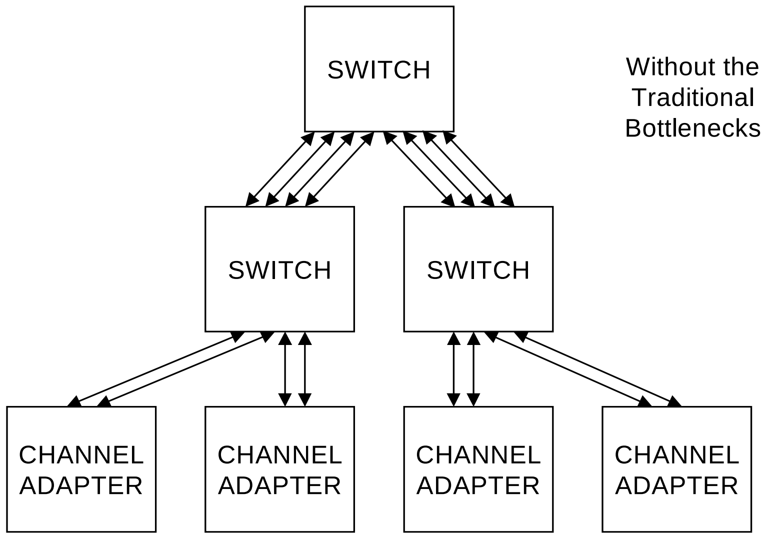
Links in a bundle may have different flight times. This potential skew is compensated for by the cell encapsulation control characters, SCD and ECD. The start of cell delimiter (SCD) indicates when the cell segment on this link of a bundle starts arriving. The end of cell delimiter (ECD) indicates when the cell segment has fully arrived. These are repeated on each link of a bundle at the beginning and end of each cell. All transactions across bundled links are synchronized by this replicated encapsulation.

Figure 30: **Rich Fabrics, Bandwidth Matching, Link Speed Matching**



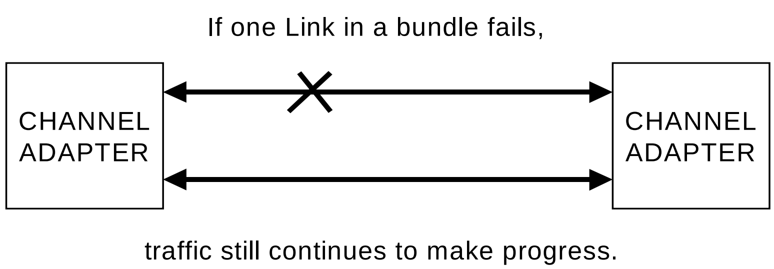
If the bundle size is consistent through the fabric from source to destination bandwidth improvement is linear. If consistent from source to destination, a two link bundle will have twice the bandwidth of a single link. Bundle size does not have to be consistent through the fabric. It is not effective for individual cells with no congestion unless the effective bandwidth of the bundle is consistent from source to destination. If the size of the bundle is not consistent through the fabric, bandwidth is constrained to that of the narrowest bundle of links. If the size of the bundle is not consistent through the fabric, switches may be forced by backpressure store and forward rather than cut-through. On the positive, this feature may be used to implement a pruned tree topology with out the usual bandwidth bottlenecks.

Figure 31: **Pruned Tree Topology**



Bundling offers failover capabilities. Redundant bundles can be set up. When one link in a bundle fails, some portion of the others are still able to carry traffic, although at a reduced bandwith capacity. This redundant capacity is the next power of two decrement.

Figure 32: **Reliability Through Redundancy**



B.1.1 Current Proposals

Auto-discovery brings links online as unbundled. Links are bundled during fabric configuration and reconfiguration. Three conditions must be met.

- Devices between aggregated links must support bundling.
- Aggregated links must be a desired configuration.
- The fabric manager can identify and successfully configure the bundle.

Cell characters are striped and reassembled in a configured order specified by the fabric manager when establishing bundled links. Ports have numbers or local reference addresses associated with them. Bundles must be assembled on a power of two port number boundary with links assembled in order from least significant to most significant consistent with the port numbers.

Links are aggregated in ascending, power of two port number order by fabric management software.

- 0,1,2,3 not 1,3,5,7
- 2,3 not 3,4

- 4,5,6,7 not 3,4,5,6

The fabric manager has the capabilities to detect, check, interact, and assist, if necessary, with configuration of bundles.

B.1.2 Work in Progress

The current issues are under discussion and will have some bearing on the specification of MLX.

- Flow control on bundles.
- Optimal striping width.
- More detail on fail-over and drop-off synchronization.
- Link ordering requirements.